VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2014 Bc. Jiří Antecký

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Aplikace metody zarovnání snímků lidaru pro lokalizaci autonomního robotu

Scan Matching Method Application for Autonomous Robot Localization

2014 Bc. Jiří Antecký

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Jiří Antecký**

Studijní program: N2649 Elektrotechnika

Studijní obor: 3901T009 Biomedicínské inženýrství

Téma: Aplikace metody zarovnání snímků lidaru pro lokalizaci autonomního robotu
Scan Matching Method Application for Autonomous Robot Localization

Zásady pro vypracování:

1. Rozbor metod pro zpracování laserových snímků.
2. Seznámení se s ICP algoritmem.
3. Návrh a realizace matematického modelu (Matlab).
4. Implementace ICP algoritmu pro možnost online nasazení (C, C++, C#, ...).
5. Provedení experimentů a testů s ICP algoritmem.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] TVARŮŽKA, Adam. *Senzorický subsystém robotu.* Ostrava, 2008. Disertační práce. Vysoká škola báňská - Technická univerzita Ostrava, Fakulta strojní.
[2] RUSINKIEWICZ, Szymon and Marc LEVOY. Efficient variants of the icp algorithm. In: *Proceedings of Third International Conference on 3-D Digital Imaging and Modeling,* Quebec City, Que, 28 May 2001-01 Jun 2001. pp.145–152. ISBN 0-7695-0984-3.
[3] SIEGWART, Roland, Illah R. NOURBAKHSH and Davide SCARAMUZZA. *Introduction to Autonomous Mobile Robots.* 2nd ed. Massachusetts: Massachusetts Institue of Technology, c2011, xvi, 453 s. Intelligent robotics and autonomous agents. ISBN 978-0-262-01535-8.
[4] BURGUERA, Antoni, Gabriel OLIVER and Juan D. TARDOS. Robust scan matching localization using ultrasonic range finders. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2-6 Aug. 2005. pp. 1367–1372. ISBN 0-7803-8912-3.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jaromír Konečný**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 6.května 2014 ..... Anlecký .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6.května 2014 ..... Anlecký .....

PODĚKOVÁNÍ

Velice rád bych poděkoval svému vedoucímu diplomové práce panu Ing. Jaromírovi Konečnému za jeho rady a připomínky při tvorbě diplomové práce, především za rady při implementaci matematického modelu do programovacího jazyka.

Také bych rád poděkoval své rodině a přítelkyni za podporu nejen při tvoření diplomové práce, ale i při celém průběhu studia.

Abstrakt

Následující diplomová práce se zabývá realizací ICP algoritmu, který se používá pro lokalizaci autonomního robota při jeho pohybu v reálném prostředí. Diplomová práce nejprve popisuje nejběžnější algoritmy, které se využívají pro zarovnání laserových snímků.

Další částí práce je vytvoření matematického modelu ICP algoritmu. K této realizaci je využit program Matlab. Poslední částí diplomové práce je vytvoření programu pro on-line nasazení ICP algoritmu a jeho použití na reálných datech, které jsou získány pomocí laserového skeneru LMS SICK 100. K této realizaci je použit programovací jazyk C#.

Klíčová slova: Zarovnání, lokalizace, algoritmus, ICP, laserový skener, LMS SICK 100, Matlab, C#

Abstract

The following master thesis is dealing with the implementation of ICP algorithm, which is used for localization of autonomous robot in real environment during movement. Master thesis the first describe the most common algorithms, which are used for laser scans aligning.

The next step of master thesis is to create mathematical model of ICP algorithm. To realization of this program is used Matlab program. The final part of master thesis is to create a program for on-line deployment and testing them on real data sets get by LMS SICK 100 laser scanner. For program realization is used C# programming language.

Keywords: Aligning, localization, algorithm, ICP, laser scanner, LMS SICK 100, Matlab, C#

Seznam použitých zkratek a symbolů

A	–	skutečná hodnota [mm]
$\mathbf{A}$	–	obecná matice bodů
APR	–	Anchor Point Relationship
ASCII	–	American Standard Code for Information Interchange {Americký Standardní Kód pro Výměnu Informací}
BSD	–	Berkeley Software Distribution
$\mathbf{C}$	–	vektor kovarianční matice
CAN	–	Controller Area Network
CLS	–	Complete Line Segment
CPU	–	Central Processing Unit
D	–	vzdálenost [mm]
DS_p	–	vzdálenostní spektrum referenčního snímku
DS_q	–	vzdálenostní spektrum aktuálního snímku
F	–	cílová (účelová) funkce
FLANN	–	Fast Library for Approximate Nearest Neighbors
GUI	–	Graphical User Interface {Grafické uživatelské rozhraní}
$\mathbf{H}(\mathbf{x}_n)$	–	hessián účelové funkce
HSM	–	Hough Scan Matching
HS_p	–	Houghovo spektrum referenčního snímku
HS_q	–	Houghovo spektrum aktuálního snímku
HT	–	Hough Transform {Houghova Transformace}
ICP	–	Iterative Closests Point
IP	–	Internet Protocol
LGPL	–	Lesser General Public License
LMS	–	Laser Measurement Systems {Laserové Měřící Systémy}
$\mathbf{M}$	–	matice množiny bodů
$\mathbf{M}_{t+1}$	–	matice nové množiny bodů
MCL	–	Monte Carlo Localization {Monte Carlo Localizace}
N	–	počet bodů
NDT	–	Normal Distributions Transform
P	–	pravděpodobnost

PSM	–	Polar Scan Matching {Polární porovnávání snímků}
QIM	–	Quadratic Interpolation Method {Kvadratická Interpolační Metoda}
$\mathbf{R}$	–	matice rotace
SLAM	–	Self Localization And Mapping
TCP	–	Transmission Control Protocol
b	–	heuristická konstanta
c	–	konstanta
$\mathbf{c}$	–	vektor těžiště
d	–	vzdálenost [mm]
d_i	–	vzdálenost bodu od překážky [mm]
$\mathbf{g}(\mathbf{x}_n)$	–	gradient účelové funkce
$\mathbf{m}_i$	–	vektor bodu množiny $\mathbf{M}$
$\mathbf{p}_i$	–	vektor bodu referenčního snímku
$\mathbf{p}_n$	–	vektor největšího spádu
p_x	–	souřadnice bodu referenčního snímku v ose x
p_y	–	souřadnice bodu referenčního snímku v ose y
$\mathbf{q}_i$	–	vektor bodu aktuálního snímku
q_x	–	souřadnice bodu aktuálního snímku v ose x
q_y	–	souřadnice bodu aktuálního snímku v ose y
r	–	poloměr [mm]
$\mathbf{t}$	–	vektor rotace
t_x	–	parametr transformace v ose x
t_{x0}	–	počáteční podmínky translace v ose x
t_y	–	parametr transformace v ose y
t_{y0}	–	počáteční podmínky translace v ose y
v_i	–	váha bodu
x	–	souřadnice v ose x
x_i	–	souřadnice bodu v ose x
$\mathbf{x}_i$	–	vektor stavových proměnných
$\mathbf{x}_n$	–	vektor parametrů transformace v n-té iteraci
$\mathbf{x}_{n+1}$	–	vektor parametrů transformace v $n+1$-té iteraci

$\mathbf{x}^*$	–	vektor bodu ostrého minima
y	–	souřadnice v ose y
y_i	–	souřadnice bodu v ose y
Δ	–	absolutní chyba [mm]
Δ_ϕ	–	změna úhlu [rad]
α	–	délka kroku
β	–	úhel dané změřené vzdálenosti [rad]
γ	–	naměřená hodnota [mm]
δ	–	relativní chyba [%]
θ	–	orientovaný úhel [rad]
ϕ	–	úhel natočení (rotace) [rad]
ϕ_0	–	počáteční úhel natočení (rotace) [rad]
$\star$	–	křížová korelace
0	–	index počáteční fáze
i	–	index i-té fáze
n	–	index n-té fáze
p	–	index referenčního snímku
q	–	index aktuálního snímku
x	–	index pro osu x
y	–	index pro osu y
ϕ	–	index úhlu ϕ

Seznam tabulek

Seznam obrázků

Obsah

1 Úvod

V mobilní robotice, která se rozšiřuje čím dál do více oblastí, je hlavním úkolem robota správně se zorientovat ve známém i neznámém prostoru. Autonomní roboti vyžadují úplnou samostatnost své navigace v tomto prostředí. Dnes se tato orientace provádí pomocí srovnání a zarovnání několika laserových snímků, díky kterým je robot schopen zjistit nejen tvar prostředí ve kterém se pohybuje, ale i svou vlastní polohu v tomto prostředí. Existuje několik možných algoritmů a metod, které toto zarovnání provádějí.

Cílem této diplomové práce je nejprve pochopit důvody lokalizace autonomního robota v daném prostředí, kde lokalizace se provádí pomocí zarovnání několika laserových snímků. Dalším cílem je nastudovat a porozumět metodám, které zprostředkovávají zarovnávání snímků a teoreticky tyto metody popsat. Jednou z možných metod používaných k lokalizaci autonomních robotů v reálném prostředí je ICP algoritmus, kterým se podrobněji zabývá tato diplomová práce.

Prvním cílem praktické části diplomové práce je vytvořit návrh matematického modelu ICP algoritmu. Tento návrh a jeho následné zrealizování je provedeno pomocí softwaru Matlab. Dalším cílem praktické části je implementace algoritmu pro online využití. Tato implementace může být provedena pomocí některého z programovacích jazyků.

Jedním z posledních cílů diplomové práce je odzkoušení matematického modelu i programu určeného pro online nasazení na reálných datech, které jsou získány pomocí laserového senzoru. Po úspěšném odzkoušení programů posledním cílem diplomové práce je zhodnocení dosažených výsledků získaných z programu vytvořeného softwarem Matlab i z programu určeného pro možnost lokalizace robota v reálném čase. Mezi tyto výsledky může být zahrnuto porovnání programů a statistické vyjádření jejich přesnosti.

2 Rozbor metod pro zpracování laserových snímků

V dnešní době je většina pohyblivých robotů vybavena laserovým snímačem. Laserový snímač měří vzdálenost mezi robotem a objekty v okolí s vysokým úhlovým rozlišením s rozsahem typicky $180°$ až $360°$. Také umožňují měření s vysokým časovým rozlišením, a to nejméně 20 snímků za sekundu. Naměřené vzdálenosti mohou být použity k rekonstrukci objektů v blízkosti prostředí robota, které je tvořeno sadou bodů. Cílem zpracování snímků je porovnat dvě sady skenů a z nich získat informaci o aktuální poloze robota. Tato operace se v literatuře označuje jako mapování. Samotné mapování je dáno 2D translací a 2D rotací. K vypočtení tohoto posunutí a otočení může být použito několik metod (algoritmů), které jsou uvedeny dále v textu diplomové práce. [1]

2.1 Algoritmus ICP

ICP (Iterative Closest Point) je algoritmus, který byl navržen Beslem a McKayem [2] pro minimalizaci rozdílů sumy čtverců vzdálenosti mezi dvěma sadami bodů. Tato minimalizace je založena na předešlém vytvoření dvojic nejbližších bodů z referenčního a aktuálního skenu. Z nalezených souřadnic těchto bodů je vypočtena transformace, která provede již zmíněnou minimalizaci rozdílu vzdálenosti. Z důvodu, že průběh minimalizace vzdálenosti mezi páry bodů se opakuje do splnění některé z podmínek pro ukončení algoritmu, je algoritmus iterační. ICP algoritmus je možné použít i pro 3D mapováni prostoru. Naopak jeho nevýhodou je větší výpočetní náročnost. Samotný průběh ICP algoritmu lze shrnout do několika po sobě jdoucích krocích, například dle [3]:

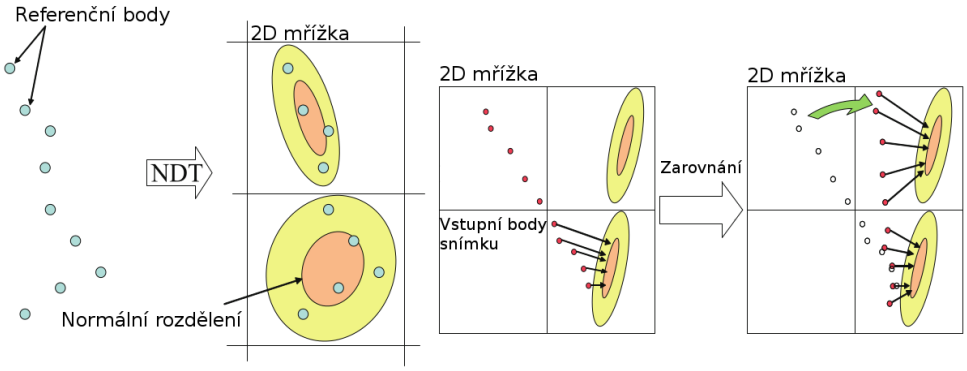
1. *Předzpracování (Preprocessing)*
2. *Přiřazení*
3. *Odmítnutí*
4. *Určení hodnoty cílové funkce*
5. *Minimalizace cílové funkce*

Tyto uvedené kroky budou podrobně popsány v další části textu diplomové práce 3.

2.2 Algoritmus NDT

Algoritmus NDT (Normal Distributions Transform), vytvořený Biberem [1], pracuje na jiném principu než ostatní SLAM algoritmy. Funkce algoritmu je taková, že rozděluje

snímané body do buněk mřížky v pravidelných intervalech a poté je celá tato mřížka aproximována normálním rozdělením. Prakticky NDT algoritmus přiřazuje body z aktuálního skenu k normálnímu rozdělení referenční mřížky. Aproximace a přiřazení jednotlivých bodů snímku je ukázáno na obrázku (obr. 1). [4]



Obr. 1 Princip NDT algoritmu

NDT algoritmus je rychlejší z důvodu, že nepotřebuje hledat pro aktuální snímek žádné referenční mřížky. Postup přiřazení aktuálního bodu k referenčnímu podle [1] lze popsat:

1. Načtení všech 2D bodů $\mathbf{q}_{i=1..N}$ v aktuálním snímku.
2. Vypočtení těžiště bodů:
$$\mathbf{c} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_q \tag{1}$$
kde N představuje počet bodů v buňce mřížky
3. Vypočtení kovarianční matice:
$$\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{q}_i - \mathbf{c}) (\mathbf{q}_i - \mathbf{c})^{\mathsf{T}} \tag{2}$$
kde $\mathbf{c}$ je těžiště bodů
4. Vypočtení pravděpodobnosti nalezení bodu v daném místě buňky:
$$P(x) = \exp\left(-\frac{(\mathbf{p}_i - \mathbf{c})^{\mathsf{T}} \mathbf{C}^{-1} (\mathbf{p}_i - \mathbf{c})}{2} \right) \tag{3}$$
kde C je kovarianční matice a
$\mathbf{p}_i$ je bod referenčního snímku

Pomocí zjištěné pravděpodobnosti algoritmus dokáže přiřadit bod do správné buňky. Každá buňka musí mít vhodné rozměry, které se volí podle velikosti jednotlivých skenů. Díky tomuto rozdělení získáme po částech spojitou dvou dimenzionální funkci, která v 2D rovině vytváří funkci hustoty pravděpodobnosti. Posledním krokem NDT algoritmu je zarovnání nového snímku s referenčním snímkem, kde při několikanásobné iteraci získáme co nejpřesnější zarovnání. [4]

2.3 Metoda APR

Jeden z dalších algoritmů, používající se pro zpracování laserových skenů, je APR (Anchor Point Relationship) algoritmus. Úkolem tohoto algoritmu je najít shodu mezi aktuálním laserovým snímkem a sadou referenčních snímků. K tomuto přiřazení se používají takzvané kotevní body (anchor points). Tyto významné body jsou podle Webera[5] rozděleny do tří skupin:

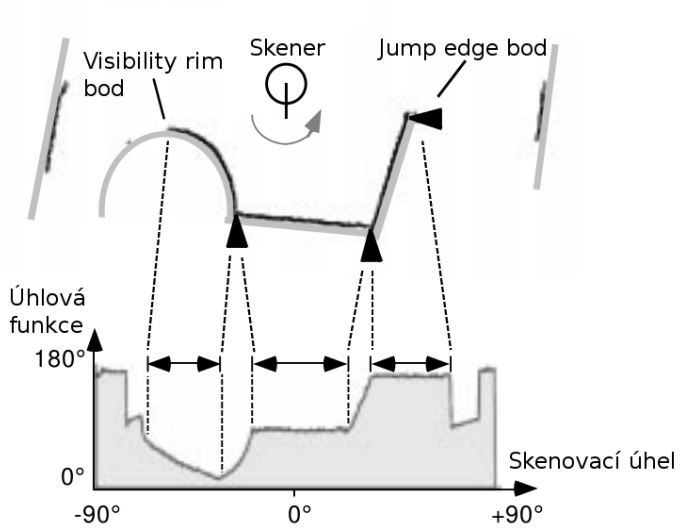
1. *Jump edge*
2. *Angle*
3. *Virtual edge*

Jump edge body jsou nejsnáze detekovatelné. Tyto body se objevují na hranicích dvou kolmých ploch při přechodu laserového paprsku z jedné plochy na druhou. Při tomto skoku se bod objeví na bližší ploše. [5]

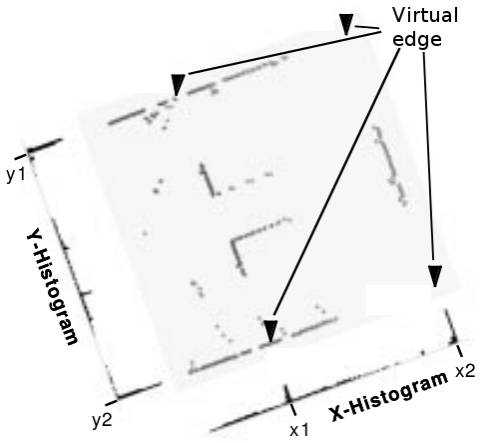
Angle (hranový) bod se objevuje při spojitém přechodu mezi dvěma plochami. Dále se také může objevit takzvaný *Visibility rim*, což je druh *jump edge* bodu, objevující se mezi přechodem kulatého předmětu a rovné plochy. Tyto tři případy kotevních bodů jsou vyobrazeny na obrázku (obr. 2). [5]

Posledním kotevním bodem v APR algoritmu je *Virtual edge* bod. Tyto body jsou vytvořeny z maximálních špiček x a y-histogramů, které představují úhlový rozdíl mezi orientací robota a úhlem svírajícím s okolní zdí. Tento druh kotevního bodu společně s x a y-histogramem uveden na obrázku (obr. 3).

Všechny tyto kotevní body pomáhají s přiřazením aktuálního laserového snímku k sadě referenčních, které obsahují kotevní body celé skenované oblasti. Určení konečné transformace mezi skeny se provádí vytvořením dvojic kotevních bodů mezi snímky. [5]



Obr. 2 Kotevní body APR algoritmu



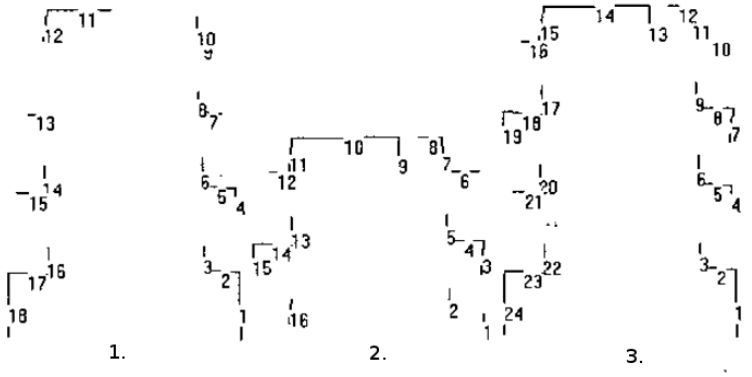
Obr. 3 Virtual edge body

2.4 Metoda CLS

Pro pravoúhlá prostředí s rovnými liniemi se hojně využívá CLS (Complete Line Segment) algoritmus. Algoritmus je založen na shodě mezi aktuálním a referenčním (globálním) snímkem. Samotné skeny se skládají z několika úseček, které jsou senzorem zaznamenány proti směru hodinových ručiček. Přiřazení aktuálního a referenčního

snímku je založeno na porovnávání vztahů jednotlivých úseček prostoru. Po následném přiřazení je možné zjistit aktuální pozici robota. [6]

Prvním úkolem algoritmu je vytvoření globální mapy snímaného prostoru. Po vytvoření globální mapy se nasnímá lokální mapa, která se sloučí s vytvořenou globální mapou prostoru. Na obrázku (obr. 4) je vyobrazen celý proces, kde obrázek 1. je vytvořená globální mapa, obrázek 2. je lokální mapa a obrázek 3. je sloučení těchto map. [6]



Obr. 4 Tvoření globální mapy

Po vytvoření globální mapy a aktuálního skenu se určuje pozice robota. Tato pozice se stanovuje pomocí přiřazení aktuálního skenu k referenčnímu. Přiřazení se provádí pomocí výpočtu pravděpodobnosti shody úseček referenčního a aktuálního snímku. Po přiřazení snímků je stanovena orientace snímku ϕ, díky které je schopno vypočítat pomocí rovnice (4) aktuální pozici robota,

$$\begin{aligned} x &= x_p - (x_q \cos(\phi) + y_q \sin(\phi)) \\ y &= y_p - (y_q \cos(\phi) + x_q \sin(\phi)) \end{aligned} \tag{4}$$

kde x_p, y_p jsou souřadnice úsečky v referenčním snímku a

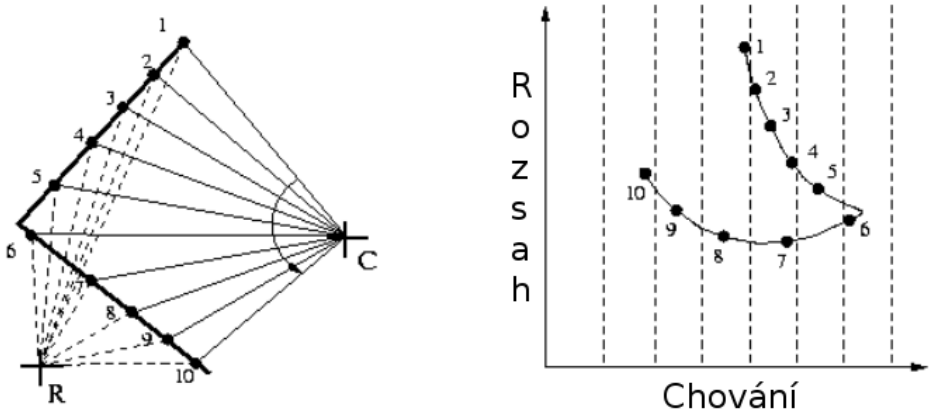
x_q, y_q jsou souřadnice úsečky v aktuálním snímku

2.5 Metoda PSM

PSM (Polar Scan Matching) metoda je založená na podobném principu jako ICP algoritmus, vytvořená v roce 2007 Diosim [12]. Stejně jako v ICP algoritmu se

porovnávají nejbližší body (suma čtverců vzdáleností) z aktuálního a referenčního snímku. Jeho hlavní výhodou oproti ICP algoritmu je ve výpočetní rychlosti nejbližších sousedů.[12]

Prvním krokem PSM algoritmu je předzpracovávání (*preprocessing)*, kde se pomocí mediánového filtrování odstraní odlehlé body z referenčního a aktuálního snímku (obr. 5).



Obr. 5 PSM Algoritmus

Druhým krokem algoritmu je promítnutí aktuálního snímku do souřadnic referenčního snímku. Třetím krokem algoritmu je odhadnutí translace. V tomto kroku PSM algoritmus je pomocí odometru změřen co nejširší úhlový rozsah. Pro zobrazené body je pak vypočítán odhad posunu podle rovnice (5):

$$\begin{bmatrix} t_x & t_y \end{bmatrix} = \text{argmin}_{x_q y_q} \left[\sum_i \mathbf{w}_i \parallel \mathbf{p}_i - \mathbf{q}_i \left(x_c, y_c \right) \parallel^2 \right] \tag{5}$$

kde $\mathbf{q}_i \left(x_q, y_q \right)$ je bod otočený o souřadnice x_q a y_q z aktuálního snímku.

Poté je pomocí kvadratické interpolační metody (QIM) vypočítána úhlová rotace aktuálního snímku k referenčnímu, kde výsledné otočení je vypočteno pomocí :

$$\phi = \phi_0 + b\Delta_\phi \tag{6}$$

kde b je heuristická konstanta a Δ_ϕ je změna možného úhlu

Toto je pouze stručný popis algoritmu, podrobná funkce algoritmu je napsána v literatuře [8, 12].

2.6 Monte Carlo lokalizace

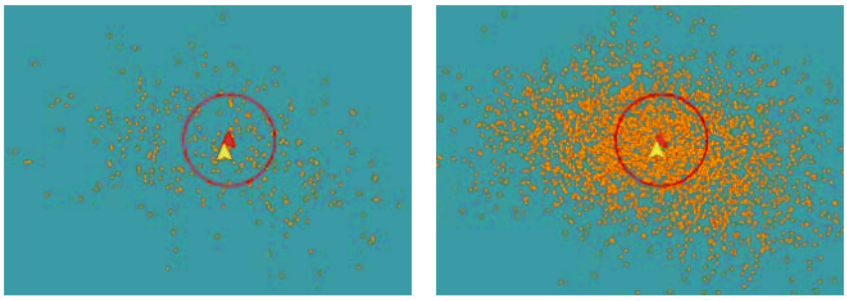
Monte Carlo lokalizace (MCL) je algoritmus, který byl v 70. letech objeven Handschinem [13]. Po dalších dvaceti letech byl nezávisle na předchozím objevu znovuobjeven a využit k určování pozice robota [14]. V tomto textu bude pouze stručně popsán hlavní princip MCL algoritmu, podrobnější popis lokalizace je uveden v [14, 15]. Principem MCL je takový, že prezentuje pravděpodobnost stavu prostředí konečnou množinou vážených bodů. V této množině $\mathbf{M}$ je každý bod určen (7):

$$m_i = (v_i, \mathbf{x_i}) \tag{7}$$

kde v_i je váha daného bodu a

$\mathbf{x_i}$ je vektor stavových proměnných

Z tohoto vyjádření vyplývá, že pravděpodobnost přítomnosti robota v určitém místě je dána váhami bodů a jejich hustotou v daném místě. Se zvyšujícím se počtem bodů v blízkosti robota se přesnost zvyšuje, viz. obrázek (obr. 6). Hlavním úkolem lokalizace je úprava této množiny bodů. Tato množina pak obsahuje informaci o pozici robota.[14]



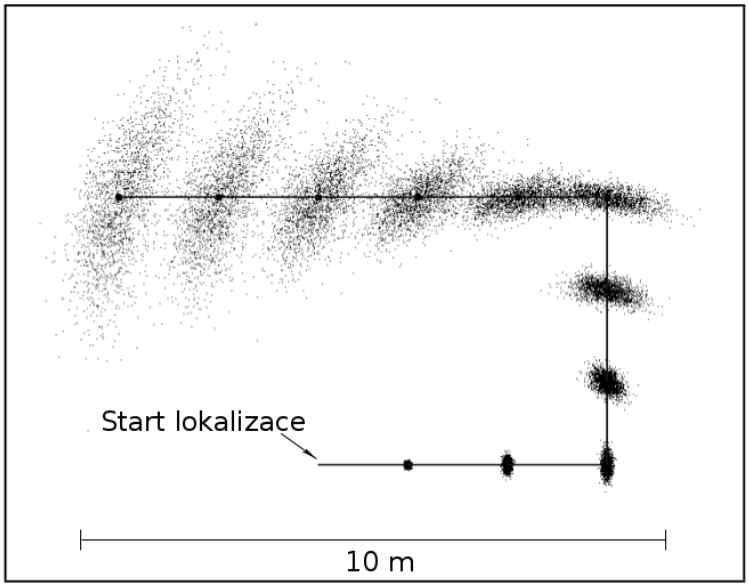
Obr. 6 Aproximace s různým počtem bodů

MCL podle [15] pracuje ve čtyřech iteračních krocích:

1. *Predikční krok*
2. *Translační krok*
3. *Korekční krok*
4. *Převzorkování*

Predikční krok využívá takzvaný přechodový model. Pohybem robota získáme novou množinu bodů $\mathbf{M}_{t+1}$ tak, že ke každému bodu m_{jt} z množiny $\mathbf{M}_t$ přidáme novou

hodnotu vektoru proměnných $\mathbf{x}_{i(t+1)}$. Tento vektor se generuje použitím přechodového modelu. Po tomto kroku se zvyšuje nejistota pozice robota (obr. 7).



Obr. 7 Zvyšující se nejistota

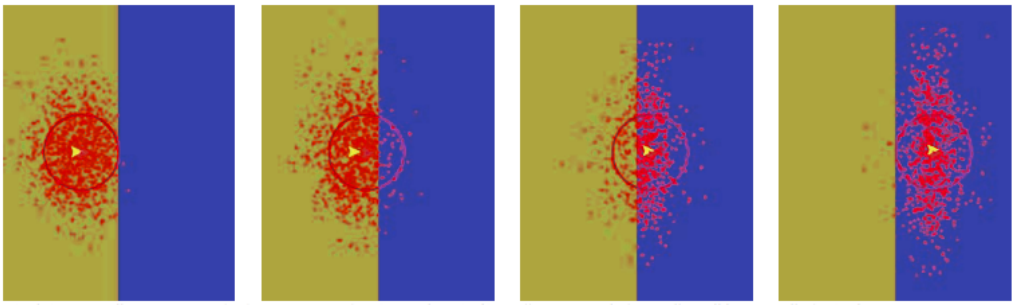
Translační krok pracuje se změnou polohy robota. U tohoto kroku však samotný pohyb robota nahradíme čtením dat ze senzoru.

Korekční krok pouze zpracovává údaje získané snímačem. Oproti predikčnímu a translačnímu kroku nijak neupracuje vektor $\mathbf{x}_j$, pouze upravuje váhy jednotlivých bodů.

Převzorkování je posledním krokem MCL algoritmu. Tato část algoritmu má za cíl z aktuální množiny bodů získat novou množinu bodů. Tato nová množina musí obsahovat body se stejnými váhami jako množina předchozí. Korelační krok a převzorkování je zobrazeno na obrázku (obr. 8), kde jednotlivé obrázky ukazují postup při opakování celého procesu.[16]

2.7 Metoda HSM

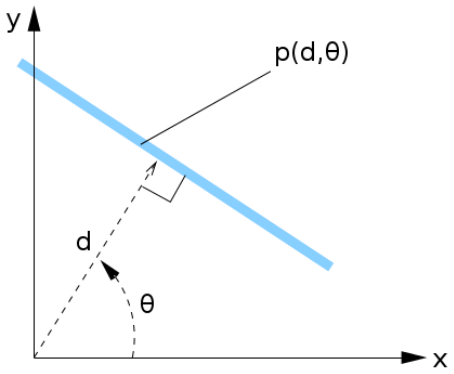
Metoda HSM (Hough Scan Matching), vytvořena Censi a kol. [7], využívá k zarovnání laserových snímků Houghovu transformaci.



Obr. 8 Princip Monte Carlo lokalizace

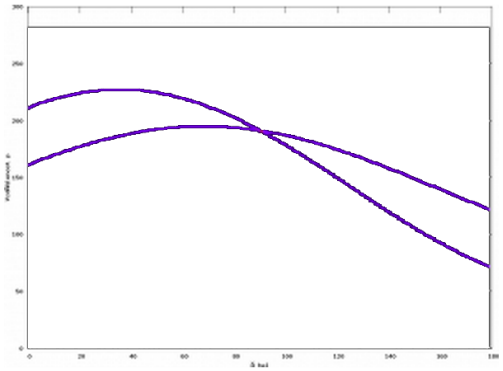
Houghova transformace je metoda, která slouží k hledání jednoduchých geometrických tvarů v obraze. U přímky definované vztahem (8) , lze určit vzdálenost přímky od počátku. Po vypočtení vzdálenosti d a velikosti orientovaného úhlu θ je možné každý bod v obraze převést na souřadnice $[d, \theta]$ (obr. 9). [7]

$$y = x\frac{x\cos(\theta)}{\sin(\theta)} + \frac{d}{\sin(\theta)} \tag{8}$$



Obr. 9 Vyjádření přímky

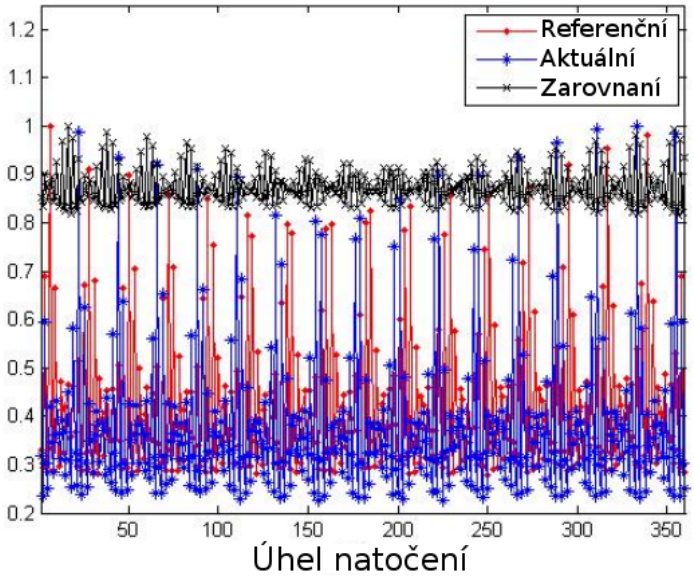
Po vytvoření tohoto bodu v obraze je umožněno bodem pronést všechny možné přímky. Tímto postupem v obraze vnikne sinusoida. Pokud hledáme parametry mezi dvěma přímkami vzniknou nám dvě protínající se sinusoidy. Vytvořený bod protnutí poté určuje hledané parametry (obr. 10).



Obr. 10 Dva body vyjádření pomocí Houghovy transformace

Oproti ICP algoritmu, HSM metoda je neiterační a neporovnává jednotlivé body prostoru, ale místo toho porovnává Houghovy spektra jednotlivých snímků. Podle [8] HSM metoda pracuje v několika krocích:

1. Vytvoření Houghova spektra pro referenční a aktuální snímek (obr. 11).



Obr. 11 Vytvořené Houghovy spektra

2. Vypočtení jejich křížové korelace (Cross-correlation).

$$(f \star g)(\phi) = \sum_{\theta} HS_q(\theta) \cdot HS_p(\theta - \phi) \tag{9}$$

kde
HS_p je Houghovo spektrum referenčního snímku a
HS_q je Houghovo spektrum aktuálního snímku

3. Určení transformačního úhlu z maxima vzájemného vztahu.

$$\phi = \text{argmax}_{\phi}(f \star g)(\phi_0) \tag{10}$$

4. Výpočet vzdálenostního spektra referenčního a aktuálního snímku.
5. Výpočet vzájemného vztahu vzdálenostních spekter.

$$DC(k) = \sum_{k} DS_q(d) \cdot DS_p(d - k) \tag{11}$$

kde
DS_p je vzdálenostní spektrum referenčního snímku a
DS_q je vzdálenostní spektrum aktuálního snímku

6. Vypočtení maximální vzdálenosti mezi referenčním a aktuálním vzdálenostním spektrem.

$$|D| = \text{argmax}_{k} DC(k) \tag{12}$$

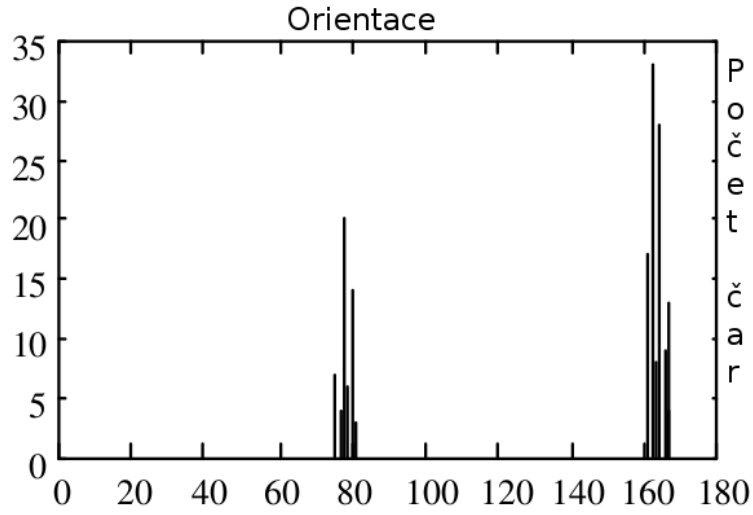
7. Vypočtení souřadnic x, y posuvu aktuálního snímku od referenčního.

$$\begin{aligned} t_x &= |D| \cos(\phi_0) \\ t_y &= |D| \sin(\phi_0) \end{aligned} \tag{13}$$

2.8 Korelační metoda na základě histogramu

Tato korelační metoda, vytvořená Weissem [9], využívá pro zarovnání snímků několik histogramů, ve kterých pomocí jejich lokálních maxim je možné zjistit změnu natočení a posuv mezi aktuálním a referenčním snímkem. [9]

U zjišťování změny natočení mezi jednotlivými snímky se po nasnímání prostoru pomocí laserového skeneru vypočítají úhly, které senzor svírá s naměřenými předměty. Poté je z naměřených úhlů vytvořen úhlový histogram, ve kterém jsou pro orientaci vyjádřeny nejvíce svírané úhly mezi snímačem a prostředím (obr. 12).

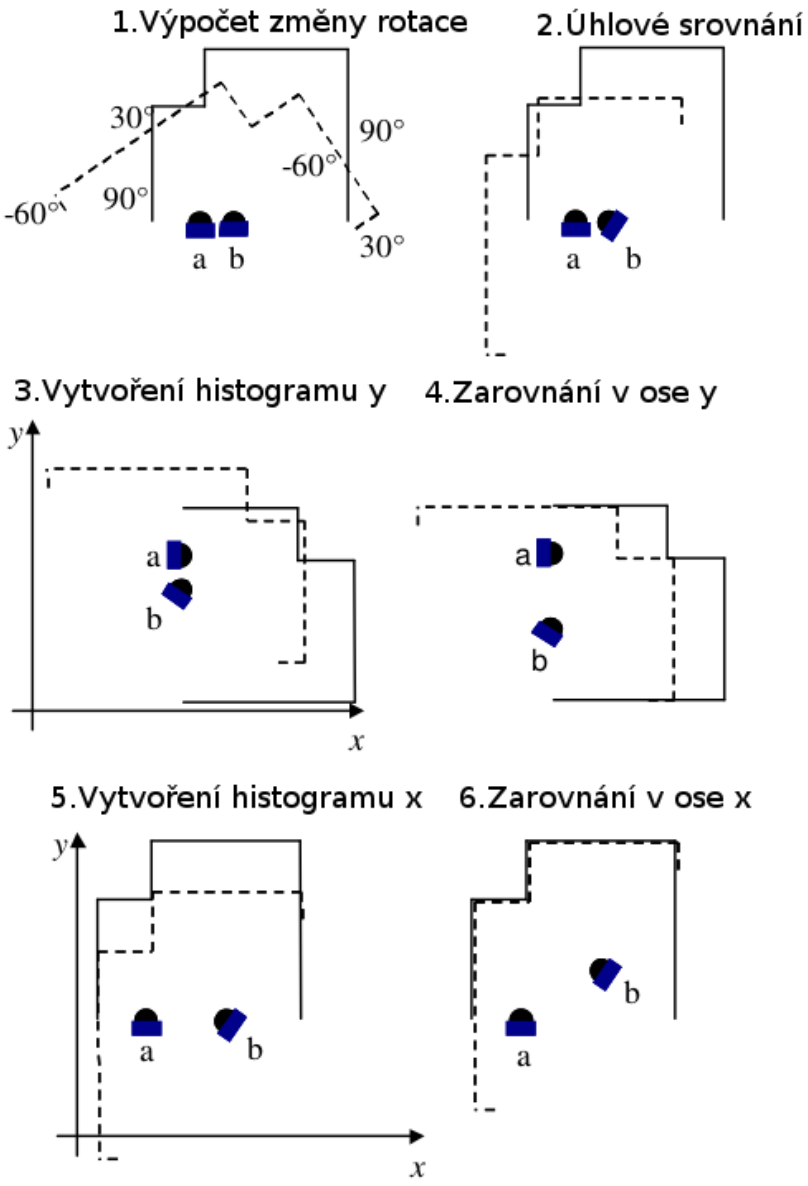


Obr. 12 Úhlový histogram

Stejný postup je pak proveden u aktuálního snímku a poté je pomocí křížové korelace určen rozdíl natočení mezi aktuálním a referenčním snímkem.[10]

$$(f \star g)(y) = \lim_{x \to \infty} \frac{1}{2x} \int_{-x}^{x} f(x)\, g(x+y)\, dx \tag{14}$$

V reálném prostředí se však snímky od sebe neliší pouze změnou natočení, ale i polohou snímače. Toto zarovnávání se provádí tak, že se nejprve odfiltrují body neobjevující se v obou typech snímků. Po této filtraci se provede výpočet změny rotace. Po určení změny rotace pomocí rovnice (14), se jednotlivé snímky úhlově srovnají. Poté se provede výpočet translace mezi snímky. Nejprve se obrazy nakloní na osu x a vytvoří se histogram pro osu y. Z něj se vypočten rozdíl mezi posunutím aktuálního a referenčního skenu v ose y a následně jsou snímky zarovnány(d). Nakonec se tento postup provede i pro osu x, které vede ke konečnému zarovnán snímků. Celý tento postup je zobrazen na obrázku (obr. 13).[11]



Obr. 13 Postup zarovnávání snímků pomocí histogramů

3 Seznámení se s ICP algoritmem

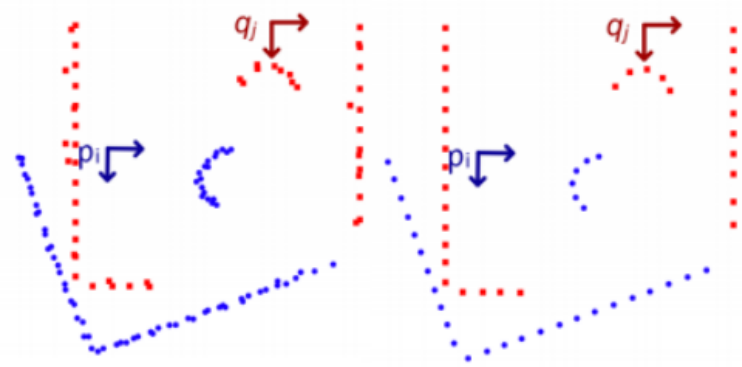
Jak již bylo zmíněno v kapitole 2, ICP algoritmus můžeme popsat v několika krocích, díky kterým je schopen zarovnat aktuální a referenční snímek. Sběr těchto snímků je umožněn pomocí laserového skeneru SICK LMS, který je popsán v kapitole 4.1.1. V této části diplomové práce jsou více rozvedeny principy jednotlivých kroků a co znamenají pro samotný ICP algoritmus.

Podle literatury [2, 3, 17] lze říci, že ICP algoritmus ve 2D prostředí hledá parametry transformace (t_x, t_y, ϕ), díky kterým je schopen minimalizovat Euklidovskou vzdálenost mezi jednotlivými body referenčního a aktuálního snímku.

Z názvu samotného algoritmu je patrné, že se jedná o takzvaný iterační algoritmus. To znamená, že jednotlivé kroky algoritmu se opakují do té doby, než je splněna (v tomto případě) jedna z podmínek optimalizační metody.

3.1 Předzpracování

Prvním krokem ICP algoritmu je takzvané předzpracování, neboli preprocessing. V tomto kroku algoritmu se provádí filtrace bodů,za účelem redukce velikosti dat souboru. Mezi tyto filtrované body můžeme zařadit například vzdálené body, které nepomáhají z lokalizaci robota v daném prostředí. Dále mohou být odfiltrovány body, které jsou nahuštěny vedle sebe, jak ze ukázáno na obrázku (obr. 14). [2, 17]



Obr. 14 Předzpracování

3.2 Přiřazení

Druhým krokem algoritmu je takzvané přiřazení (matching). U přiřazení se hledají nejlépe korespondující body mezi aktuálním a referenčním snímkem. Tento krok je z celého algoritmu nejvíce časově náročný. Existuje několik metod pro nalezení korespondujících bodů mezi snímky, kde v této diplomové práci jsou použity metody:

- Metoda řešení hrubou silou (brute force)
- k-d strom

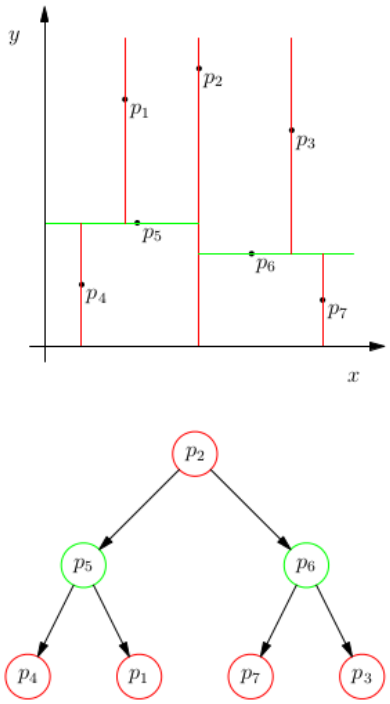
3.2.1 Metoda řešení hrubou silou

Metoda řešení hrubou silou je nejjednodušší metoda pro hledání nejbližších sousedů mezi aktuálním a referenčním snímkem. Funguje tak, že se vypočítá vzdálenost mezi bodem referenčního snímku a kandidáty ze snímku aktuálního. Po vypočtení vzdálenosti se vybere ten, který je bodu z referenčního snímku nejblíže. Největší nevýhoda této metody je její kvadratická složitost $\mathcal{O}\left(N^2\right)$. Díky této složitosti je tento algoritmus nevhodný pro snímky s větším počtem bodů. Z tohoto důvodu se tato metoda v diplomové práci nevyužívá. [17]

3.2.2 k-d strom

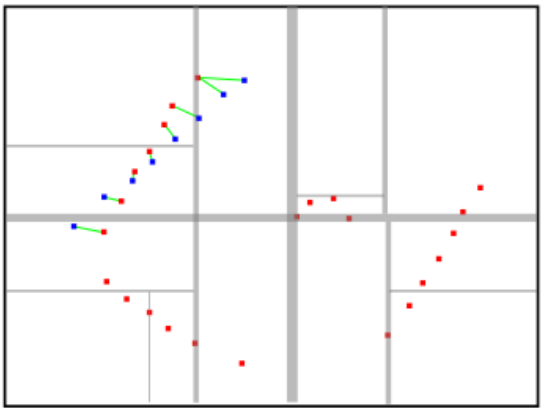
Jednou z další možnosti hledání nejbližšího souseda je za pomocí takzvaného k-dimenzionální stromu (k-d tree). Jeho princip je takový, že rozdělí množinu bodů $\mathbb{P}$, která je v určitém dimenzionálním zobrazení (například 2D zobrazení), na několik polorovin tak, že každý bod je umístěn ve vlastním prostoru. Toto rozdělení prostoru na podprostory se provádí pomocí výpočtu mediánu daného prostoru. Ve 2D prostoru je každý bod z množiny $\mathbb{P}$ dán souřadnicemi (p_x, p_y). Při rozdělení prostoru pomocí k-d stromu se nejprve vyhledá medián osy x , kterým se prostor rozdělí na dva podprostory. Poté se v každém podprostoru vyhledá medián osy y a podprostor se rozdělí na další dvě části. Takto se pokračuje dále až do té doby, dokud není splněna podmínka o konečném počtu listů. Tento celý postup je ukázán na obrázku (obr. 15). [17]

U hledání nejbližších sousedů mezi referenčním a aktuálním snímkem se na jednotlivé podprostory rozdělí snímek referenční. Poté se na takto rozdělený snímek položí snímek aktuální a hledají se korespondující body v jednotlivých rovinách (obr. 16). Hlavní výhoda hledání nejbližšího souseda pomocí k-d stromu je jeho



Obr. 15 Hledání nejbližších sousedů pomocí kd stromu

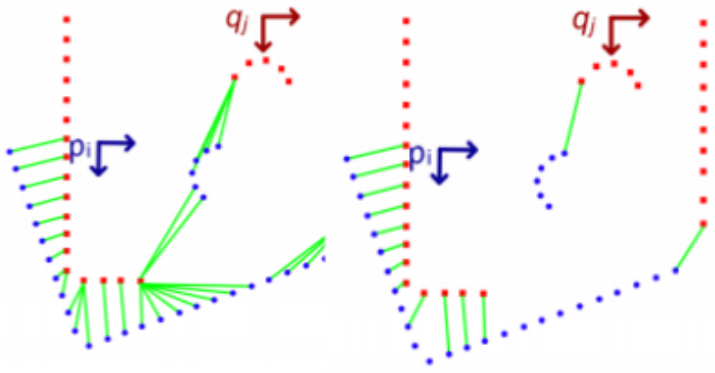
logaritmická složitost $O(\log_2 N)$, díky které je výpočetní rychlost několikanásobně vyšší než u metody řešení hrubou silou. [17]



Obr. 16 Hledání nejbližších sousedů mezi referenčním a aktuálním snímkem

3.3 Odmítnutí

Ve třetím kroku, který se nazývá odmítnutí (rejecting), se provádí eliminace páru bodů referenčního a aktuálního snímku. Odmítnutí je prováděno na základě výpočtu vzdálenosti mezi těmito dvěma páry, kdy se ponechává pár s nejmenší vzdáleností. Tento proces je vyobrazen na obrázku (obr. 17). Dále také u částečně překrytých snímků můžou být odstraněny body, které zahrnují okrajové vrcholy snímků. Tomuto odstranění se říká odmítnutí vrcholů hran a je použito spíše ve 3D prostředí. [17]



Obr. 17 Odmítnutí bodů

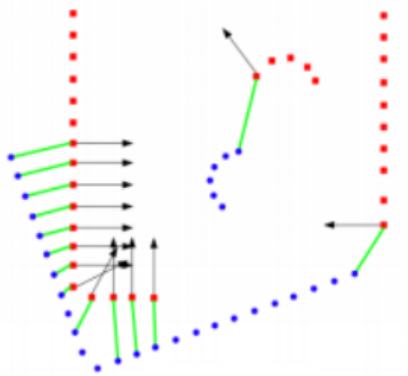
3.4 Určení hodnoty cílové funkce

V kroku určení hodnoty cílové funkce se vypočte hodnota účelové funkce mezi páry bodů aktuálního a referenčního snímku, kde určení je možné vidět na obrázku (obr. 18).

Tato hodnota účelové funkce je minimalizována každou iterací algoritmu. O kroku minimalizace cílové funkce je napsáno více v podkapitole 3.5. Samotná hodnota účelové funkce je pak použita k:

- *Point to point* minimalizaci, nebo
- *Point to plane* minimalizaci.

U *Point to point* minimalizace je cílová funkce vyjádřena sumou čtverců vzdáleností (Euklidovská norma) bodů aktuálního snímku k bodům snímku referenčního. Cílová



Obr. 18 Určení hodnoty cílové funkce

funkce pak může být vyjádřena pomocí rovnice (15):

$$F\left(t_x, t_y, \phi\right) = \sum_{i=1}^{N} \parallel \left(\mathbf{R}\mathbf{q}_i + \mathbf{t}\right) - \mathbf{p}_i \parallel^2 \tag{15}$$

přičemž ve 2D prostoru $\mathbb{R}^3 \rightarrow \mathbb{R}$ a kde:

$\mathbf{p}_i \in (p_x, p_y)$ a jedná se o souřadnice bodu referenčního snímku,

$\mathbf{q}_i \in (q_x, q_y)$ jsou souřadnice bodu aktuálního snímku,

$\mathbf{R}$ je matice rotace a

$\mathbf{t}$ je vektor translace (posuvu).

Matici rotace $\mathbf{R}$ můžeme pak vyjádřit pomocí rovnice (16):

$$\mathbf{R} = \begin{bmatrix} \cos\left(\phi\right) & -\sin\left(\phi\right) \\ \sin\left(\phi\right) & \cos\left(\phi\right) \end{bmatrix} \tag{16}$$

Point to plane minimalizace je naopak vyjádřena sumou vzdáleností bodů aktuálního snímku k tečně roviny, ve které jsou umístěny body snímku referenčního. Toto tvrzení lze matematicky zapsat pomocí rovnice (17):

$$F\left(t_x, t_y, \phi\right) = \sum_{i=1}^{N} \parallel \left(\mathbf{R}\mathbf{q}_i + \mathbf{t}\right) \cdot \mathbf{n}_k - \mathbf{p}_i \parallel^2 \tag{17}$$

kde $\mathbf{n}_k$ vyjadřuje zarovnané tečné normály v k-tém bodu referenčního snímku.

3.5 Minimalizace cílové funkce

V posledním kroku ICP algoritmu se iteračně provádí minimalizace cílové funkce. Abychom mohli minimalizovat sumu čtverců rozdílu mezi aktuálním a referenčním snímkem, musíme vypočítat parametry transformace t_x, t_y, ϕ. Tyto parametry lze určit pomocí optimalizačních metod, které obecně spadají do kategorie nelineárního programování. Jednou z často využívaných metod je Newtonova metoda. [19, 18]

Newtonova metoda se často využívá pro řešení soustavy nelineárních rovnic. Nicméně ji lze využít také pro optimalizaci, kde v tomto případě má tvar rovnice (18),

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \left[\mathbf{H}\left(\mathbf{x}_n\right)\right]^{-1} \mathbf{g}\left(\mathbf{x}_n\right), n \geq 0 \tag{18}$$

kde α je délka kroku (u klasické metody je $\alpha = 1$),

$\mathbf{g}\left(\mathbf{x}_n\right)$ je vypočtený gradient v n-té iteraci a

$\mathbf{H}\left(\mathbf{x}_n\right)$ je vypočtený Hessián v n-té iteraci.

Dále také Newtonovu metodu lze použít k nalezení extrému funkce. Postačující podmínky pro určení ostrého minima v bodě $\mathbf{x}^*$ jsou:

1. $\mathbf{g}\left(\mathbf{x}^*\right) = \mathbf{0}$
2. $\mathbf{H}\left(\mathbf{x}^*\right)$ je pozitivně definitní

Matice je pozitivně definitní za předpokladu, že se jedná o symetrickou matici a také pokud má kladná vlastní čísla. Pokud jsou splněny tyto podmínky, potom $\mathbf{x}^*$ je bodem ostrého lokálního minima funkce *F*. V této diplomové práci se využívá Newtonovy metody k určení minima funkce F, která je uvedena v rovnici (19) vycházející z rovnice (18),

$$F = \sum_{i=1}^{N} \left[\left(q_{x,i}\cos\left(\phi\right) - q_{y,i}\sin\left(\phi\right) + t_x - p_{x,i}\right)^2 + \left(q_{x,i}\sin\left(\phi\right) + q_{y,i}\cos\left(\phi\right) + t_y - p_{y,i}\right)^2\right] \tag{19}$$

kde q_{xi} a q_{yi} jsou souřadnice *i*-tého bodu z původní matice,

ϕ je úhel rotace,

p_{xi} a p_{yi} jsou souřadnice bodu referenčního snímku a

t_x a t_y je posuv ve směru x a y.

U klasické Newtonovy metody je délka kroku $\alpha = 1$. Pro urychlení konvergence se α mění v každém iteračním kroku programu. Více o této volbě kroku je uvedeno v podkapitole 3.5.3.

3.5.1 Výpočet gradientu

Pro výpočet gradientu se vychází z rovnice (19), který se derivuje podle jednotlivých parametrů t_x, t_y, ϕ. Z toho plyne, že gradient je vektorem o prvcích g_1, g_2, g_3.

$$\mathbf{g}\left(t_x, t_y, \phi\right) = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} \frac{\partial F}{\partial t_x} \\ \frac{\partial F}{\partial t_y} \\ \frac{\partial F}{\partial \phi} \end{bmatrix} \tag{20}$$

Jednotlivé složky gradientu se pak rovnají:

$$g_1 = \frac{\partial F}{\partial t_x} = 2\sum_{i=1}^{N}\left(x_i \cos\left(\phi\right) - y_i \sin\left(\phi\right) + t_x - b_{xi}\right) \tag{21}$$

$$g_2 = \frac{\partial F}{\partial t_y} = 2\sum_{i=1}^{N}\left(y_i \cos\left(\phi\right) + x_i \sin\left(\phi\right) + t_y - b_{yi}\right) \tag{22}$$

$$\begin{aligned} g_3 = \frac{\partial F}{\partial \phi} = 2\sum_{i=1}^{N} & \left(x_i \cos\left(\phi\right) - y_i \sin\left(\phi\right) + t_x - b_{xi}\right) \\ & \cdot \left(-x_i \sin\left(\phi\right) - y_i \cos\left(\phi\right)\right) \\ & \cdot \left(y_i \cos\left(\phi\right) + x_i \sin\left(\phi\right) + t_y - b_{yi}\right) \\ & \cdot \left(-y_i \sin\left(\phi\right) + x_i \cos\left(\phi\right)\right) \end{aligned} \tag{23}$$

3.5.2 Výpočet Hessiánu

Stejně jako u výpočtu gradientu, vychází výpočet Hessiánu z rovnice (19), kde ještě navíc po první derivaci se provede druhá derivace funkce podle rovnice (24), přičemž se jedná o symetrickou matici.

$$\mathbf{H}\left(t_x, t_y, \phi\right) = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 F}{\partial t_x^2} & \frac{\partial^2 F}{\partial t_x \partial t_y} & \frac{\partial^2 F}{\partial t_x \partial \phi} \\ \frac{\partial^2 F}{\partial t_y \partial t_x} & \frac{\partial^2 F}{\partial t_y^2} & \frac{\partial^2 F}{\partial t_y \partial \phi} \\ \frac{\partial^2 F}{\partial \phi \partial t_x} & \frac{\partial^2 F}{\partial \phi \partial t_y} & \frac{\partial^2 F}{\partial \phi^2} \end{bmatrix} \tag{24}$$

Vypočtené prvky matice pak jsou uvedeny v následujících rovnicích:

$$H_{11} = \frac{\partial^2 F}{\partial t_x^2} = 2 \tag{25}$$

$$H_{12} = \frac{\partial^2 F}{\partial t_x \partial t_y} = 0 \tag{26}$$

$$H_{13} = \frac{\partial^2 F}{\partial t_x \partial \phi} = 2 \sum_{i=1}^{N} \left[-x_i \sin(\phi) - y_i \cos(\phi) \right] \tag{27}$$

$$H_{21} = \frac{\partial^2 F}{\partial t_y \partial t_x} = 0 \tag{28}$$

$$H_{22} = \frac{\partial^2 F}{\partial t_y^2} = 2 \tag{29}$$

$$H_{23} = \frac{\partial^2 F}{\partial t_y \partial \phi} = 2 \sum_{i=1}^{N} \left[-y_i \sin(\phi) + x_i \cos(\phi) \right] \tag{30}$$

$$H_{31} = \frac{\partial^2 F}{\partial \phi \partial t_x} = 2 \sum_{i=1}^{N} \left[-x_i \sin(\phi) - y_i \cos(\phi) \right] \tag{31}$$

$$H_{32} = \frac{\partial^2 F}{\partial \phi \partial t_y} = 2 \sum_{i=1}^{N} \left[-y_i \sin(\phi) + x_i \cos(\phi) \right] \tag{32}$$

$$\begin{aligned} H_{33} = \frac{\partial^2 F}{\partial \phi^2} = 2 \sum_{i=1}^{N} & -t_x x_i \cos(\phi) + t_x y_i \sin(\phi) \\ & + b_{xi} x_i \cos(\phi) - b_{xi} y_i \sin(\phi) \\ & - t_y x_i \sin(\phi) - t_y y_i \cos(\phi) \\ & + b_{yi} x_i \sin(\phi) + b_{yi} y_i \cos(\phi) \end{aligned} \tag{33}$$

3.5.3 Volba délky kroku

Jedna z nejjednodušších metod pro volbu vhodné délky kroku α je takzvané *Armijovo zpětné vyhledávání*, které vyžaduje splnění nerovnice (34). Jeho hlavním cílem je snížení počtu iterací nutných k nalezení parametrů cílové funkce. [19]

$$F(\mathbf{x}_n + \alpha \mathbf{p}_n) \leq F(\mathbf{x}_n) + c\alpha \mathbf{p}_n^T \mathbf{H} F(\mathbf{x}_n) \tag{34}$$

kde $\mathbf{x}_n$ jsou parametry transformace v n-té iteraci,

α je délka kroku,

$\mathbf{p}_n$ je vektor největšího spádu, který je dán rovnicí $\mathbf{p}_n = -\mathbf{H}^{-1}\mathbf{g}$ a

c je konstanta, která podle literatury [18] $c = 10^{-4}$.

4 Návrh a realizace matematického modelu

Prvním bodem praktické části diplomové práce je vytvoření návrhu a realizaci matematického modelu ICP algoritmu. K vytvoření matematického modelu byl použit program Matlab. Pomocí získaných teoretických znalostí ICP algoritmu sepsáných v kapitole 3, byl sestrojen vývojový diagram (obr. 19) pro zrealizování matematického modelu. Nejprve byly pro správný chod programu využívány přídavné toolboxy programu Matlab, konkrétně statistický a optimalizační toolbox.

Po celkovém zprovoznění programu byla snaha se této závislosti na toolboxech zbavit, buď nahrazením vlastního kódu, nebo využitím externí knihovny. Toto odstranění závislosti na přídavných toolboxech programu Matlab bylo velmi důležité, z důvodů další implementace algoritmu vhodného pro online nasazení, které bylo navrženo a zrealizováno pomocí programovacího jazyku C#. Jednotlivé bloky vývojového diagramu jsou popsány v následných podkapitolách.

4.1 Výběr referenčního, aktuálního snímku a filtrace bodů

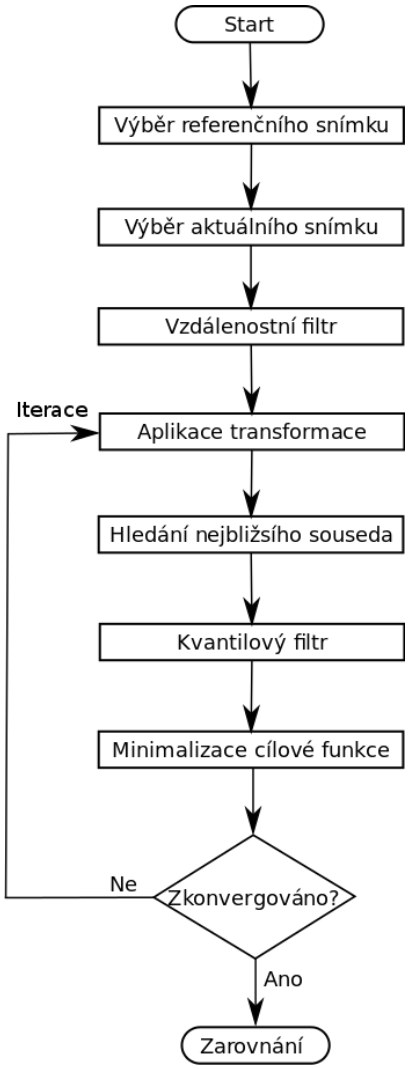
Prvními bloky vývojového diagramu jsou výběry aktuálního a referenčního snímku. Samotné snímky jsou pořízeny pomocí laserového senzoru SICK LMS 100, který je více popsán v podkapitole 4.1.1. Podle nastavení senzor v celém svém rozsahu (až $270°$) měří vzdálenost k nejbližší překážce. Pomocí vyhodnocovacího programu v počítači, se kterým senzor komunikuje přes wi-fi připojení, se nasbírané data prezentující celý snímek uloží do souboru s námi požadovaným formátem. V případě této diplomové práce je vybrán formát textového dokumentu (.txt) kvůli snadné manipulaci jak v prostředí programu Matlab, tak i pomocí programovacího jazyka C#. Takto vytvořené data pak jsou prezentovány v jednom řádku. Jelikož každý datový soubor obsahuje 1080 hodnot vzdáleností, můžeme podle údajů senzoru uvedených v (tab. 1) zjistit, že úhlové rozlišení je $0,25°$.

Aby bylo možné tyto data graficky prezentovat, musejí být určeny souřadnice x, y každého bodu ze souboru. Tyto souřadnice každého bodu jsou vypočteny pomocí rovnice (36),

$$\begin{aligned} x_i &= d_i \cos(\beta_i) \\ y_i &= d_i \sin(\beta_i) \end{aligned} \tag{35}$$

kde

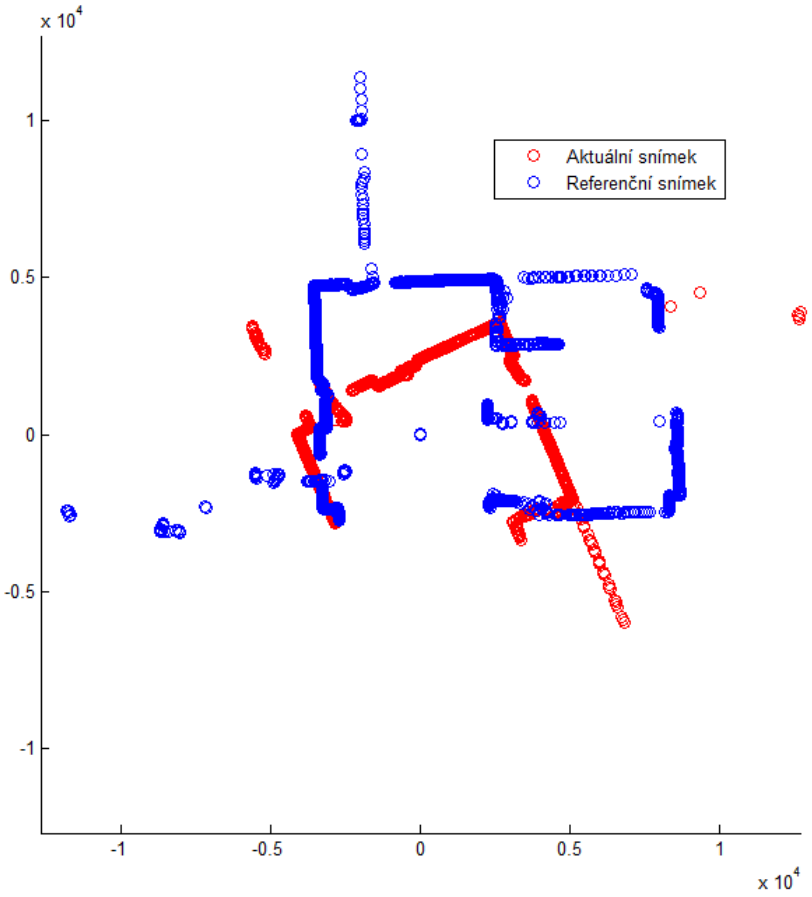
d_i vzdálenost získaná senzorem od překážky a



Obr. 19 Vývojový diagram matematického modelu

β_i úhel dané změřené vzdálenosti.

Pomocí těchto dvou rovnic se určí souřadnice každého bodu z dat získaných senzorem. Poté už bylo možné pomocí funkce programu Matlab *plot* vykreslit referenční a aktuální snímek (obr. 20). Výběr referenčního a aktuálního skenu se provádí odkomentováním příslušného textového souboru s daty a následným zakomentováním předešlého souboru, kde tyto soubory jsou načítány na začátku kódu programu.



Obr. 20 Vykreslení aktuálního a referenčního snímku

Dalším blokem vývojového diagramu je vzdálenostní filtr, který má za úkol odfiltrovat příliš vzdálené body od počátku, podle nastavené hodnoty v programu, která lze libovolně měnit. V této diplomové práci je vzdálenostní filtr nastaven na hodnotu 6 metrů, která se osvědčila jako optimální.

4.1.1 Laserový senzor SICK LMS 100

Laserový senzor LMS 100 firmy SICK (obr. 21) je širokopásmový detekční skener, díky kterému je možno pomocí optického paprsku snímat svůj okolní prostor. Princip laserového senzoru je takový, že se přímo měří vzdálenost od snímače, nejen tedy jestli vyslaný odražený paprsek byl příjmut nebo ne. Samotné měření vzdálenosti od snímače

se provádí podle měření doby od vypuštění paprsku, do doby jeho příjmu. Laserový snímač totiž nevysílá paprsek stále, ale v pravidelných intervalech. Mezi těmito intervaly vnitřní elektronika senzoru počítá uplynulý čas mezi vysláním a příjmem laserového paprsku, a podle toho určí vzdálenost od dané překážky.

Obr. 21 Laserový senzor SICK LMS 100

Měřící rozsah senzoru je až $270°$, kde senzor má možnost krokování po $0,25°$ nebo po $0,50°$. Samotný sběr informací získaných senzorem je možné uskutečnit pomocí linky RS-232, sběrnice RS-285, CAN sběrnice nebo pomocí Ethernetu. Jednotlivé parametry skeneru jsou pak uvedeny v tabulce (tab. 1).

Tab. 1 Vlastnosti laserového senzoru SICK LMS 100

Vlastnosti	
Provozní dosah	20 m
Skenovací frekvence	25 Hz nebo 50 Hz
Rozsah skenu	max. 270°
Uhlové rozlišení	$0,25°$ nebo $0,50°$
Odezva	20 ms nebo 40 ms
Datová komunikace	RS-232, RS-285, CAN bus, Ethernet
Napájení	$10,8$ V DC až 30 V DC
Rozměry	105 mm x 102 mm x 152 mm

4.2 Aplikace transformace

Dalším krokem vývojového diagramu matematického modelu ICP algoritmu je aplikace transformace. Tato afinní transformace vychází z rovnice pro účelovou funkci (15). Lze tedy pomocí této rovnice získat matici bodů o souřadnicích x, y,

$$\begin{aligned} x &= q_x \cos(\phi) - q_y \sin(\phi) + t_x \\ y &= q_y \cos(\phi) + q_x \sin(\phi) + t_y \end{aligned} \tag{36}$$

kde

q_x, q_y jsou souřadnice bodu aktuálního snímku,

t_x, t_y souřadnice posuvu (translace) a

ϕ je úhel rotace.

Na základě takto vytvořené transformační matice je možné vytvořit matici bodů pro dané parametry t_x, t_y, ϕ.

4.3 Hledání nejbližšího souseda, kvantilový filtr

Velmi důležitým krokem vývojového diagramu, a také samotného ICP algoritmu je hledání nejbližšího souseda mezi aktuálním a referenčním snímkem. Jak již bylo zmíněno v kapitole 3.2, existuje několik metod pro hledání korespondujících bodů mezi snímky. V této diplomové práci je využit pro tento účel takzvaný k-d strom, který je taktéž podrobněji popsán v kapitole 3.2.

Přídavný statistický toolbox programu Matlab obsahuje funkci *knnsearch*, která umožňuje hledání korespondujících bodů mezi dvěma mraky dat. Tato funkce má logaritmickou složitost a ke každému bodu z referenčního snímku najde indexy nejbližšího souseda z odhadované matice bodů. Jak již bylo zmíněno, v diplomové práci je snaha zbavit se jakýchkoliv závislostí na přídavných toolboxech, a proto funkce *knnsearch* je nahrazena vlastní funkcí *flann_search* využívající externí knihovnu FLANN, viz. kapitola 4.3.1.

Po vyhledání nejbližšího souseda (sousedů) je zařazen do vývojového diagramu kvantilový filtr. Tento filtr má za úkol odstranit body, které jsou od daného bodu referenčního snímku daleko a nechceme je použít v dalších krocích algoritmu. Tento filtr se jmenuje kvantilový, protože se zadává hodnota kolik procent (kolika procentní kvantil) seřazených bodů podle vzdálenosti chceme odfiltrovat a kolik jich chceme

zanechat. V matematickém modelu je hodnota filtru *dist_quantil_cutoff* nastavena na 0,5, čili je nastaven kvantil na 50% (medián).

Program Matlab s využitím statistického toolboxu obsahuje funkci *quantile*, která byla v průběhu času nahrazena vlastní funkcí *quan_apx*. Tato vlastní funkce není přímo rovnocenná kvantilu, ale její výpočet je rychlejší. Funkce pouze jednotlivé body snímku seřadí podle jejich vzdálenosti od senzoru (od nejkratší po nejdelší) a vynásobí ji námi zadanou hodnotou kvantilu. Tím se vypočte hodnota vzdálenosti, která je hraniční pro filtraci.

4.3.1 FLANN knihovna

FLANN knihovna, vytvořená dvojicí Muja a Lowe [20], slouží k rychlému vyhledávání nejbližších sousedů mezi jednotlivými snímky, zvláště ve vícedimenzionálním prostoru. Tato opensource knihovna, vydávána pod BSD licencí, je napsána v programovacím jazyku. Dále také obsahuje rozhraní pro použití v programovacím jazyku Python, C a softwaru Matlab. Za povšimnutí stojí, že pro jazyk C# není dostupné rozhraní pro práci s knihovnou. Více o této problematice je napsáno v kapitole 5. Knihovna obsahuje několik různých druhů algoritmů pro vyhledání nejbližších sousedů, jako například:

- *kdtreesingle* používající klasického k-d stromu s volitelným počtem listů
- *kdtree*, který vytváří jeden nebo více randomizovaných k-d stromů
- *kmeans* hledající nejbližšího souseda přes takzvaný *clustering*

V této diplomové práci je využit typ k-d stromu *kdtreesingle*, který s nastavením parametrů podle tabulky (tab. 2) nejvíce odpovídá chování funkci statistického toolboxu programu Matlab. Jednotlivými nastavenými parametry jsou:

- *algorithm* určuje, jaký algoritmus pro vyhledání nejbližších sousedů bude použit.
- *checks* omezuje maximální hloubku stromu, která bude prohledána při hledání nejbližšího souseda. Pokud $check = -1$ se hledá bez limitů.
- *eps* parametr pro aproximativní hledání nejbližších sousedů za předpokladu, že $eps > 0$.
- *leaf_max_size* parametr určuje maximální počet listů zamezující dalšímu větvení stromu (*bucket size*).

Tab. 2 Parametry pro vyhledání nejbližšího souseda

Parametr	Hodnota parametru
algorithm	kdtreesingle
checks	-1
eps	0,0
leaf_max_size	32

4.4 Minimalizace cílové funkce

Jednou z poslední částí vývojového diagramu matematického modelu je samotná minimalizace cílové funkce. Podle teorie z kapitoly 3.5, je potřeba vypočítat parametry dané transformace, a tím minimalizovat Euklidovskou normu (suma čtverců rozdílu vzdáleností mezi aktuálním a referenčním snímkem).

Program na svém začátku potřebuje znát počáteční parametry účelové funkce. Aby s větší pravděpodobností došlo k zarovnání snímků, je zadáno hned několik možných počátečních podmínek. Tím vznikne několik účelových funkcí s různými počátečními podmínkami. Každá účelová funkce má počáteční parametry $t_{x0}, t_{y0} = 0$ a liší se v parametru ϕ_0 s hodnotami, které se v této diplomové práci osvědčily jako optimální:

$$\phi_0 = \begin{bmatrix} -\frac{1}{2}\pi \\ -\frac{1}{5}\pi \\ -\frac{1}{10}\pi \\ 0 \\ \frac{1}{10}\pi \\ \frac{1}{5}\pi \end{bmatrix} \tag{37}$$

Díky těmto hodnotám vznikne šest cílových funkcí s jinými počátečními parametry. K zarovnání snímků se použije ta, která má nejmenší konečnou sumou čtverců rozdílu mezi snímky. Při použití více účelových funkcí se zvyšuje pravděpodobnost zarovnání, avšak narůstá také výpočetní doba minimalizace.

Dále také bylo řečeno v kapitole 3.5, že k určení parametrů transformace je využita Newtonova optimalizační metoda, viz kapitola 4.4.1. U této metody jsou pro program také důležité optimalizační podmínky. Tyto podmínky zaručují, že iterační minimalizace cílové funkce bude ukončena, pokud funkce splňuje alespoň jednu z podmínek. Tyto podmínky jsou:

- *OptTol* podmínka, při které dojde k zastavení minimalizace cílové funkce, pokud je maximální hodnota gradientu pod nastavenou hodnotou:

$$\max |\mathbf{g}(\mathbf{x}_n)| < OptTol \tag{38}$$

- *RelTolX* podmínka, při které dojde k zastavení minimalizace cílové funkce, pokud maximální relativní změna hledaných parametrů klesne pod nastavenou mez:

$$\max \left| \frac{\mathbf{x}_{n+1} - \mathbf{x}_n}{\mathbf{x}_n} \right| < RelTolX \tag{39}$$

- *RelTolFun* podmínka, při které dojde k zastavení minimalizace cílové funkce, když relativní změna cílové funkce je pod nastavenou hodnotou:

$$\left| \frac{F(\mathbf{x}_{n+1}) - F(\mathbf{x}_n)}{F(\mathbf{x}_n)} \right| < RelTolFun \tag{40}$$

Dále pak v matematickém modelu kromě těchto tří hodnot jsou nastaveny další parametry, které lze libovolně měnit. Nastavené hodnoty těchto parametrů jsou společně uvedeny s hodnotami konvergenčních podmínek v (tab. 3):

- *MaxIter* parametr určující maximální počet iterací, které může optimalizační metoda udělat.
- *ArmijoEnabled* pokud je tento parametr nastaven na hodnotu 1, využívá se k nalezení vhodné délky kroku v Newtonově metodě Armijova zpětného vyhledávání. Pokud je však hodnota tohoto parametru nastavená na hodnotu 0, délka kroku $\alpha = 1$.
- *ArmijoAlphaMax* tento parametr udává maximální hodnotu, se kterou se spouští Armijovo zpětné vyhledávání pro nalezení vhodné délky kroku.

4.4.1 Newtonova metoda v programu Matlab

Optimalizační toolbox programu Matlab obsahuje funkci *fminunc*, která provádí Newtonovu optimalizační metodu. Po vytvoření funkčního programu s využitím tohoto toolboxu byla snaha se této závislosti zbavit a byla vytvořena vlastni funkce *fminnew*, která funkci *minFunc* nahrazuje.

Z obecného tvaru Newtonovy metody (rovnice (18)) je patrné, že je potřeba k vytvoření vlastní funkce vypočítat gradient a hessián cílové funkce (rovnice (19)). Vlastní výpočet gradientu a hessiánu je již zmíněn v podkapitolách 3.5.1 a 3.5.2.

Tab. 3 Parametry Newtonovy metody

Parametr	Hodnota parametru
optTol	$1 \cdot 10^{-5}$
relTolX	$1 \cdot 10^{-3}$
relTolFun	$5 \cdot 10^{-5}$
maxIter	200
armijoEnabled	1
armijoAlphaMax	4

Z obecné rovnice Newtonovy metody je také jasné, že je potřeba vypočítat inverzní matici hessiánu. Pro tuto funkci se využívá příkazu *inv*, obsaženou ve statistickém toolboxu. Aby se opět odstranila závislost na přídavném toolboxu, musí se znát obecný tvar inverzní matice definovaná rovnicí (41) vycházející z [21],

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & D & G \\ B & E & H \\ C & F & I \end{bmatrix} \tag{41}$$

kde jednotlivé prvky matice jsou vypočteny pomocí prvků Hessovy matice:

$$\mathbf{H}^{-1} = \frac{1}{\det(\mathbf{H})} \begin{bmatrix} H_{22}H_{33} - H_{23}H_{23} & H_{13}H_{23} & -H_{13}H_{22} \\ H_{23}H_{13} & H_{11}H_{33} - H_{13}H_{13} & -H_{11}H_{23} \\ -H_{22}H_{13} & -H_{11}H_{23} & H_{11}H_{22} \end{bmatrix} \tag{42}$$

Ke správnému výpočtu inverzní Hessovy matice je nutné také určit determinant matice. Toho je možné určit pomocí programu Matlab pomocí funkce *det*, která vychází ze *Sarrusova pravidla* pro určení determinantu 3x3 matice (43).[21]

$$\begin{aligned} \det(\mathbf{A}) = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} &= A_{11}A_{22}A_{33} + A_{12}A_{23}A_{32} \\ &+ A_{13}A_{21}A_{32} - A_{13}A_{22}A_{31} \\ &- A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33} \end{aligned} \tag{43}$$

Díky tohoto pravidla je možné vypočítat determinant i pro Hessovu matici. Jelikož v Hessově matici jsou prvky H_{11} a H_{21} nulové, lze rovnici pro výpočet determinantu zjednodušit na rovnici (44).

$$\det(\mathbf{H}) = H_{11}H_{22}A_{33} - H_{13}H_{22}H_{31} - H_{12}H_{21}H_{33} \tag{44}$$

Po tomto určení determinantu Hessovy matice je možné vypočítat inverzní matici hessiánu. Posledním krokem k vytvoření funkce pro Newtonovu metodu je určit vhodnou délku kroku α. Tato volba vhodné délky kroku je provedena pomocí Armijova zpětného vyhledávání popsaného v podkapitole 3.5.3. Metoda se používá ke snížení počtu iterací Newtonovy optimalizační metody potřebné k nalezení parametrů cílové funkce.

4.5 Konvergence

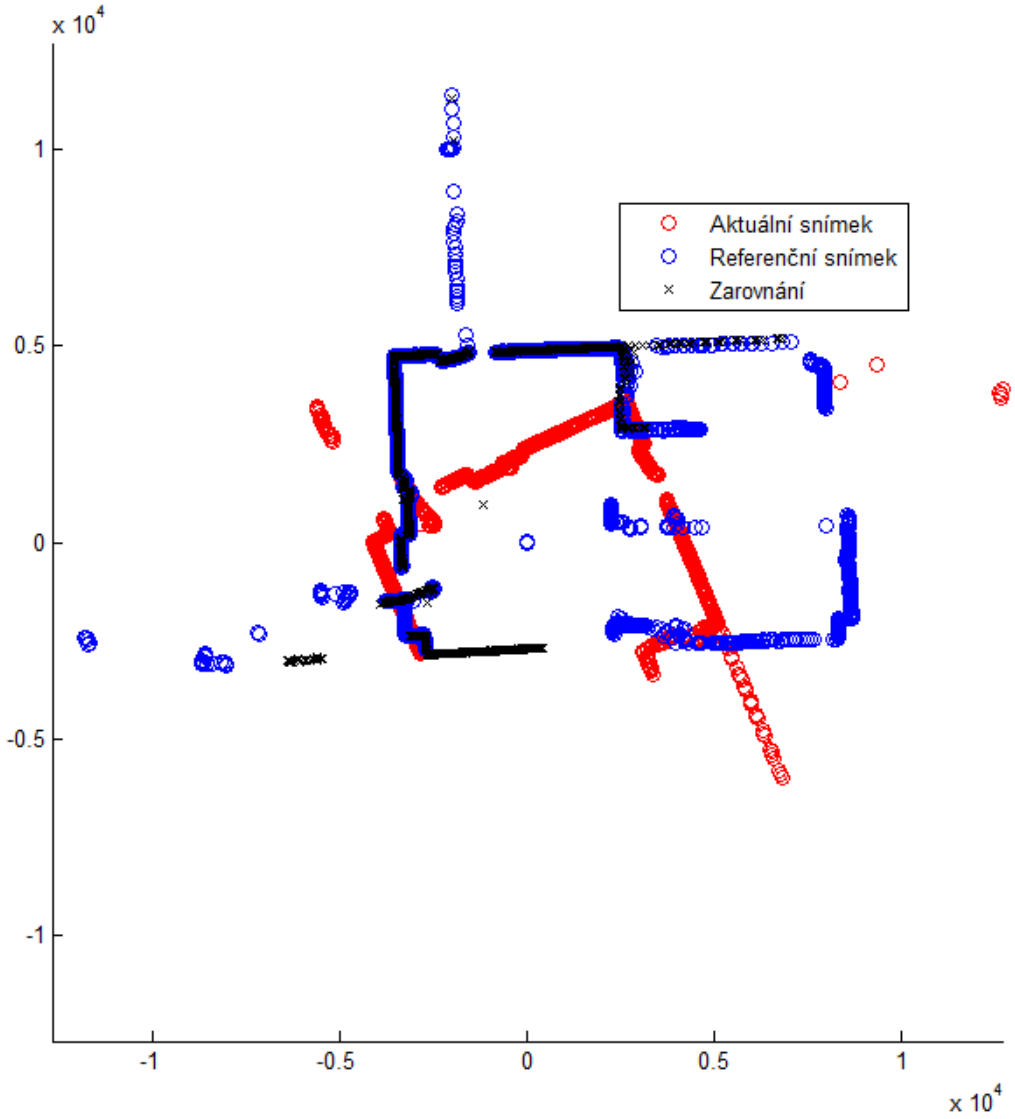
Posledním blokem vývojového diagramu je zarovnání, ve kterém se kontrolují ukončovací podmínky. U všech ze šesti počátečních podmínek cílových funkcí se provádí aplikace transformace, hledání nejbližšího souseda a minimalizace cílové funkce do té doby, dokud není splněna alespoň jedna z konvergenčních podmínek. Po výpočtech se použije ta cílová funkce s takovými počátečními podmínkami, která má nejmenší sumu rozdílů odchylek. V programu Matlab je celý tento běh programu vypsán do příkazového okna (obr. 22).

```
Dosazena RelTolFun < 5.00e-005 v 22. iteraci
Dosazena RelTolFun < 5.00e-005 v 16. iteraci
Dosazena RelTolFun < 5.00e-005 v 7. iteraci
Dosazena RelTolFun < 5.00e-005 v 17. iteraci
Dosazena RelTolX < 1.00e-003 v 36. iteraci
Dosazena RelTolFun < 5.00e-005 v 10. iteraci

Vysledky nastrelu:
   Nastrel       f(x)          tx          ty         phi
         1  1.63e+007      546.54    -128.15       -3.52
         2  1.18e+007      529.86    1205.08       -2.04
         3  9.11e+007      222.79    -795.04       -0.65
         4  3.04e+007    -584.14      846.82        0.58
         5  3.43e+004   -1187.14      975.16        1.20
         6  4.71e+007     -69.63      294.17        0.00

Nejlepsi nastrel: 5
        tx          ty         phi
  -1187.14      975.16        1.20

Elapsed time is 0.263353 seconds.
```

Obr. 22 Vypsané parametry při zarovnání snímků

Z vypsaných hodnot lze zjistit, která optimalizační podmínka u všech šesti cílových funkcí s různými počátečními podmínkami transformace byla splněna, a také v jaké

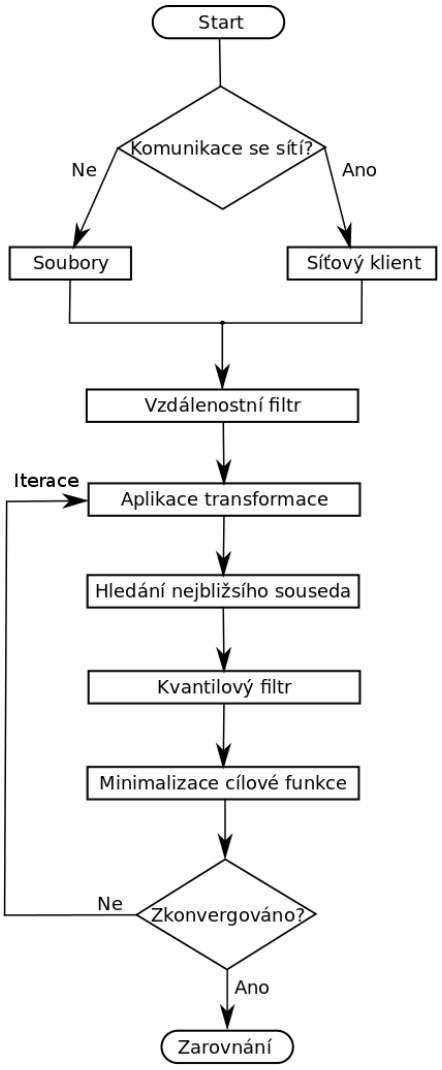
iteraci programu k tomuto splnění došlo. Dále jsou zde zobrazeny výsledné parametry transformace u cílových funkcí a jaký z těchto možných výsledků je nejlepší. Jako posledním prvkem vypsaným v příkazovém okně je konečný čas zarovnání. Také je kromě vypsaných hodnot v příkazovém okně graficky znázorněno samotné zarovnání referenčního a aktuálního snímku (obr. 23).



Obr. 23 Zarovnání aktuálního a referenčního snímku

5 Implementace ICP algoritmu pro online nasazení

Jedním z posledního úkolu diplomové práce je implementovat ICP algoritmus, který by byl vhodný pro lokalizaci robota v reálném čase. Tato implementace je vytvořena pomocí programovacího jazyka C#. Software Matlab je problematičtější pro tuto implementaci použít, protože z důvodu většího zpoždění často nevyhovuje požadavkům pro online nasazení. Stejně jako v kapitole 4 je nejprve navržen vývojový diagram celého algoritmu (obr. 24).



Obr. 24 Vývojový diagram algoritmu pro online nasazení

V tomto diagramu je největším rozdílem oproti vývojovému diagramu matematického modelu možnost volby mezi prací se soubory, nebo prací se síťovým klientem. Obě dvě možnosti jsou podrobněji popsány v příslušných kapitolách.

Zbylé jednotlivé kroky algoritmu pracují velmi podobně jako u matematického modelu. Jelikož program vytvořený v prostředí Matlab nevyužívá žádné přídavné toolboxy, je tedy možné při implementaci algoritmu pomocí programovacího jazyka vycházet přímo z něj. Dále má také matematický model s programem vytvořeným jazykem C# společné vypsání parametrů transformací, doby výpočtu a vybráním účelové funkce s nejlepšími počátečními podmínkami do příkazového řádku programu.

Stejně jako v matematickém modelu se pro nalezení nejbližšího souseda používá externí knihovna FLANN. Jak bylo řečeno v kapitole 4.3.1, knihovna je naprogramována v jazyku C++, přičemž pro práci v prostředí C# je nutné vytvořit rozhraní (*wrapper*) pro komunikaci s touto knihovnou. Autor toto rozhraní nedodává, a proto bylo nutné v rámci této diplomové práce jej vytvořit.

Velmi důležitým rozdílem mezi implementacemi Matlabu a jazyka C# je možnost paralelního výpočtu účelových funkcí s různými počátečními podmínkami. V Matlabu je tento výpočet prováděn za sebou po jednotlivých účelových funkcích. Zde je možné vypočítat parametry transformace u všech účelových funkcí najednou, a tím urychlit výpočet. K vlastní paralelizaci se používá vícevláknový přístup, přičemž jednotlivá vlákna jsou získána z fondu vláken (třída *ThreadPool*), které poskytuje běhové prostředí .NET. Výsledky vlastního zrychlení zarovnání snímků jsou uvedeny v kapitole 6.1.

5.1 Práce se soubory

U bloku vývojového diagramu práce se soubory, se postupuje podobně jako v programu Matlab. Nejprve se načte textový soubor, získaný pomocí vyhodnocovacího programu nebo pomocí programu vlastního, s pevně nastaveným počtem bodů. Pro grafické znázornění, stejně jako v kapitole 4.1, se pomocí rovnic ze vzdálenosti bodů ze souboru vypočítají souřadnice x,y.

5.2 Práce se síťovým klientem

Komunikace mezi senzorem a programem je zajištěna díky síťovému klientu. Tento klient komunikuje se senzorem pomocí TCP. Více o architektuře protokolu je možné

najít v [23]. Data vysílané senzorem mají podobu ASCII znaků. Při komunikaci mezi senzorem a počítačem, senzor vysílá 3 typy paketů:

1. *sRA STlms*
2. *sEA LMDscandata*
3. *sSN LMDscandata*

První dva druhy pouze kontrolují komunikaci a připravenost senzoru na posílání dat. Třetím typem paketu je už samotný datový paket, který začíná hlavičkou s obecnými informacemi o senzoru. Z dokumentace k senzoru [24] je zřejmé, že hlavička paketu je ukončena bajtem *0x03*. Po tomto bajtu jsou v paketu obsaženy naměřené data, které jsou ukončeny bajtem *0x02*. Následně vytvořený datový paket je poslán do bufferu, kde jsou jednotlivé pakety tvořící sken zkompletovány. Tyto data mají již podobu načítaných skenů ze souborů. [24]

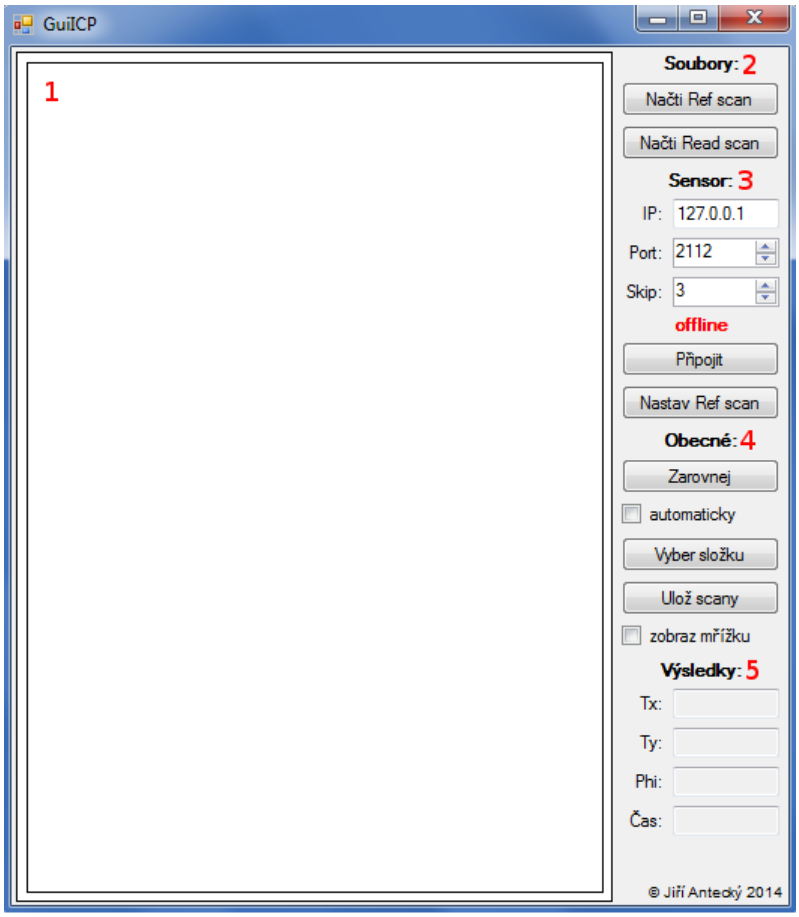
Síťový klient je realizován pomocí vícevláknového asynchronního přístupu. Aby se zabránilo zahlcení programu senzorem, předem zvolený počet datových paketů zahazuje. Tento přístup je nezbytný, neboť senzor SICK LMS 100 podle podle uživatelské příručky [24] posílá data co 120 ms a vytvořený ICP algoritmus zarovná snímky za delší dobu.

5.3 Grafické uživatelské rozhraní

Program pro online nasazení umožňuje vybírat z několika možných funkcí (práce se soubory/online, výběr snímků atd.), a proto je vhodné vytvořit grafické uživatelské rozhraní (GUI). Toto rozhraní i s celým programem je vytvořeno ve vývojovém prostředí *Microsoft Visual Studio 2010*. Pomocí editoru a externí knihovny pro GUI ZedGraph, volně stažitelná z [25] pod licencí LGPL, je vytvořeno uživatelské rozhraní (obr. 25), které lze rozdělit na pět části:

1. *Grafické okno pro vykreslení*
2. *Práce se soubory*
3. *Online komunikace se senzorem*
4. *Obecné funkce*
5. *Výsledky lokalizace*

Grafické okno slouží k vykreslení aktuálního, referenčního i zarovnaného snímku. Toto okno je společné jak při práci se soubory, tak i při práci se síťovým klientem.



Obr. 25 GUI pro ovládání programu

Část uživatelského rozhraní *Práce se soubory* obsahuje dvě tlačítka pro výběr aktuálního a referenčního snímku. Po stisknutí tlačítek se zobrazí okno s výběrem možných snímků, které musí být ve formátu *.txt* a také musí splňovat podmínku o daném počtu bodů. Tento počet bodů je nastaven stejně jako v matematickém modelu na hodnotu 1080 a při jejím nesplnění není možné soubor načíst.

Online komunikace se senzorem slouží k připojení a následné komunikaci s laserovým senzorem. V této části se zadává IP adresa senzoru a příslušný port, kde po zadání správných hodnot a stisknutí tlačítka *Připojit* začne probíhat komunikace mezi senzorem a programem. Po stisknutí tlačítka se v grafickém okně zobrazí aktuální snímek a režim *offline* se přepne do pozice *online*. Dále pomocí tlačítka *Nastav Ref Scan* se určí referenční snímek, se kterým se bude pracovat. Jako poslední tato část obsahuje

možnost volby parametru *Skip* udávající, kolik datových packetů poslaných senzorem se má přeskočit, než dojde k jejich vykreslení.

Obecné funkce obsahují tlačítka společné pro práci se soubory i pro síťovou komunikaci. Pro zarovnání snímků je nejdůležitější tlačítko *Zarovnej*, které provádí samotné zarovnání snímků. Dále je zde možnost zatrhnutí automatického zarovnání, které spouští zarovnání při každém novém aktuálním snímku. Je tedy nezbytné při online nasazení. V obecných funkcích je také možné zobrazit mřížku pro lepší orientaci. Také je zde tlačítko na výběr složky pro uložení snímku. Tlačítkem *Ulož scany* se vytvoří textové soubory od aktuálního, referenčního ale i zarovnaného snímku a uloží se do vybrané složky. Tyto textové soubory mají název podle data uložení a typu snímku. Prakticky to vypadá jako *rok_měsíc_den_hodina_minuta_vteřina_TypScanu.txt*, kde *TypScanu* může nabývat hodnot *ref*,*read* nebo *refEst*.

Výsledky lokalizace je posledním prvkem grafického rozhraní. V této části se zobrazují parametry posuvu senzoru mezi snímky (jak translační, tak i rotační) a také je zde uveden čas výpočtu těchto parametrů.

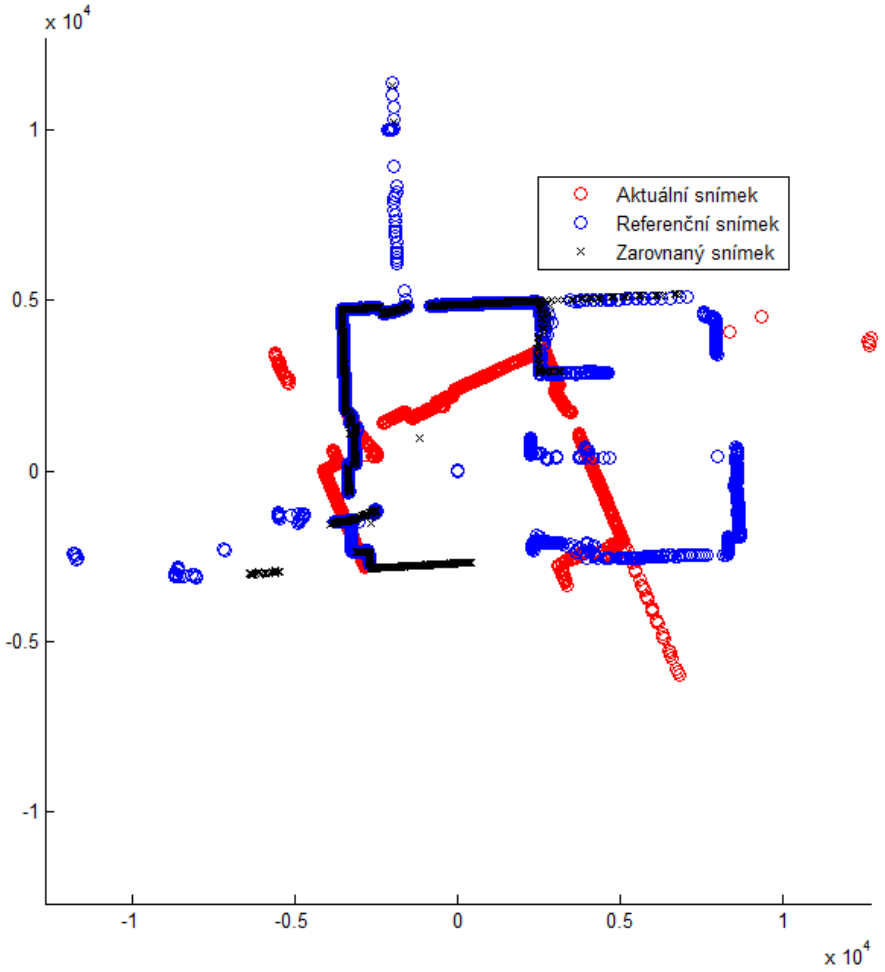
6 Testování programů a zhodnocení dosažených výsledků

Poslední částí diplomové práce je testování programů a zhodnocení dosažených výsledků ICP algoritmu jak z matematického modelu, tak z programu pro online nasazení. Pro získávání výsledků je použita měřící soustava (obr. 26), která obsahuje laserový senzor LMS 100 propojený s wi-fi routerem ethernetovým kabelem pro online komunikaci. Dále také soustava obsahuje napájecí vedení pro senzor a router. Hlavním výsledkem celé diplomové práce je vytvoření funkčního programu pro zarovnání snímků využívající ICP algoritmus a následné grafické zobrazení. Další výsledky představují:

- Porovnání rychlosti zarovnání
- Výpočet absolutní a relativní chyby měření
- Odzkoušení robustnosti algoritmu

Obr. 26 Měřící soustava

Výsledkem matematického modelu je grafické znázornění zarovnání referenčního a aktuálního skenu (obr. 27). Součástí výsledku je vypsání parametrů transformace a doby trvání výpočtu do příkazového okna programu, viz. kapitola 4.5.

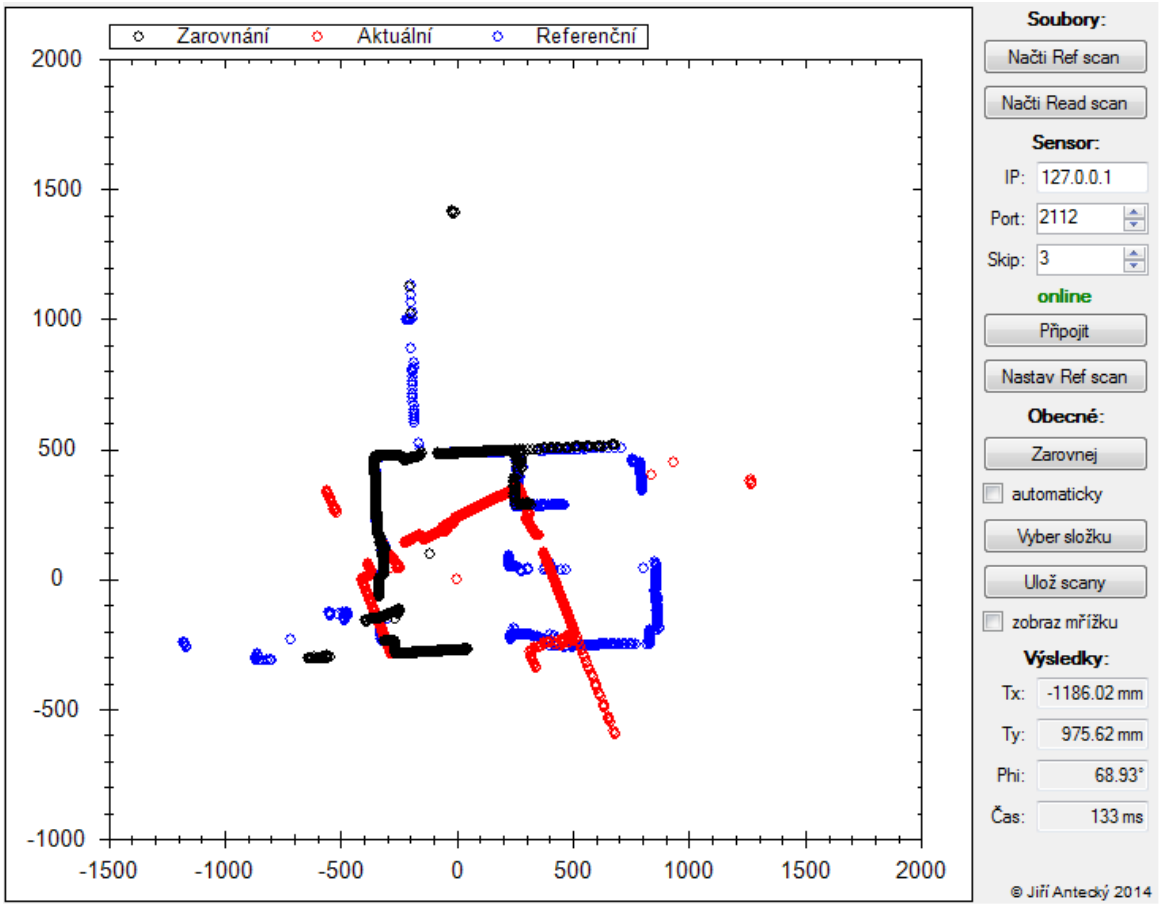


Obr. 27 Zarovnání snímků v matematickém modelu

V programu pro online nasazení vytvořeného pomocí jazyka C# je grafické znázornění obdobné (obr. 28). Výpis parametrů transformace a doba výpočtu jsou jednak vypsány v grafickém rozhraní a také jsou obsaženy v příkazovém řádku programu (obr. 29).

6.1 Porovnání rychlosti zarovnání

Při porovnání rychlosti zarovnání aktuálního a referenčního snímku v obou programech se postupovalo tak, že se při online zarovnávání ukládaly jednotlivé snímky. Poté byly



Obr. 28 Zarovnání snímků v programu pro online nasazení

snímky zarovnány pomocí matematického modelu a zjistila se doba potřebná k nalezení parametrů transformace. Při tomto porovnání se zjistilo, že matematický model, implementovaný v prostředí Matlab, je při výpočtu transformace minimálně dvakrát pomalejší než program vytvořený pomocí programovacího jazyka C#, viz. (tab. 4). Toto je dáno paralelizací výpočtu parametrů transformací při několika různých počátečních podmínkách. Důležité je také poznamenat, že doba zarovnání je velmi závislá na výkonu vyhodnocujícího počítače a jeho počtu CPU.

```
Dosazena RelTolFun < 5,00e-005 v 7. iteraci
Dosazena RelTolX < 1,00e-003 v 9. iteraci
Dosazena RelTolFun < 5,00e-005 v 18. iteraci
Dosazena RelTolFun < 5,00e-005 v 10. iteraci
Dosazena RelTolX < 1,00e-003 v 18. iteraci
Dosazena RelTolX < 1,00e-003 v 37. iteraci

Elapsed time is 0,133000 seconds.

Vysledky nastrelu:
   Nastrel        f(x)          tx          ty         phi
         1   1,73e+007      538,73    -142,67       -3,52
         2   1,24e+007      513,36    1227,91       -2,04
         3   9,25e+007      215,55    -774,48       -0,65
         4   2,88e+007    -784,04     964,00        0,73
         5   3,52e+004   -1186,02     975,62        1,20
         6   4,68e+007     -69,88     275,23        0,00

Nejlepsi nastrel: 5
        tx          ty         phi
  -1186,02      975,62        1,20
```

Obr. 29 Vypsané parametry zarovnání

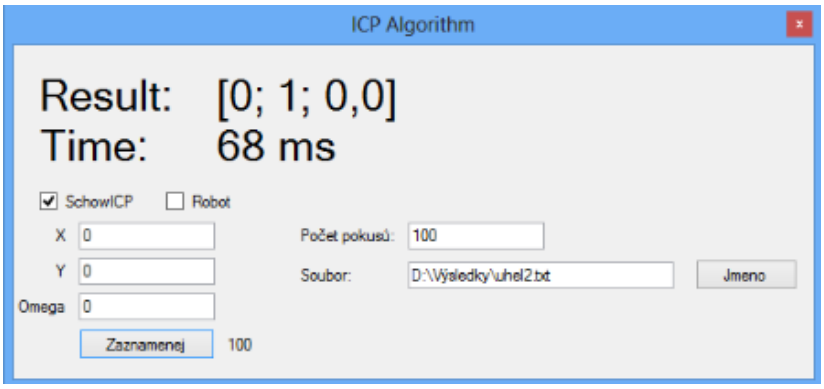
Tab. 4 Porovnání rychlostí programů

Snímek	Program Matlab [ms]	Program C# [ms]	C# oproti Matlabu [%]
1.	266	133	50
2.	211	113	53
3.	162	69	42
4.	259	109	42
5.	195	98	50
6.	187	111	59

6.2 Absolutní a relativní chyba měření

Důležitým výsledkem pro určení přesnosti lokalizace senzoru při posuvu je výpočet absolutní a relativní chyby měření. K jejímu určení byl prováděn posuv soustavy v ose y (posuv dopředu/dozadu) v rozsahu 2 metrů, s krokem 5 centimetrů a změnu natočení v hodnotách $\phi = (0°, 45°, 90°, 180°)$. Pomocí vyhodnocovacího programu (obr. 30) bylo provedeno 100 měření vzdálenosti a natočení vypočítané programem. Tyto hodnoty byly použity k výpočtu chyb měření.

U měření chyb vzdálenosti se zjistilo, že po určité délce posuvu (cca $1,5$ m) přestává algoritmus uspokojivě zarovnávat snímky. Po nastavení nového referenčního snímku



Obr. 30 Vyhodnocovací program algoritmu

algoritmus opět pracuje jak má. Tímto však vzniká integrační chyba měření. Tato chyba může být odstraněna optimalizací algoritmu (úprava počátečních podmínek účelových funkcí, změnou nastavení použitých filtrů atd.), avšak toto již není náplní diplomové práce. Stanovené hodnoty absolutní a relativní chyby měření při translačním i rotačním posuvu jsou uvedeny v tabulce (tab. 5) společně s 95% intervalem spolehlivosti. K výpočtu jsou použity rovnice (45) a (46), získané z [22]. K určení chyb při posuvu je využit 99% kvantil dat, z důvodů odstranění odlehlých hodnot měření, které vznikly nežádoucím pohybem senzoru. Na obrázku (obr. 31) jsou uvedeny krabicové grafy pro relativní chybu posuvu a rotace skenerem.

$$\Delta = |A - \gamma| \tag{45}$$

$$\delta = \frac{\Delta}{|A|} \cdot 100\% \tag{46}$$

kde Δ je absolutní chyba,

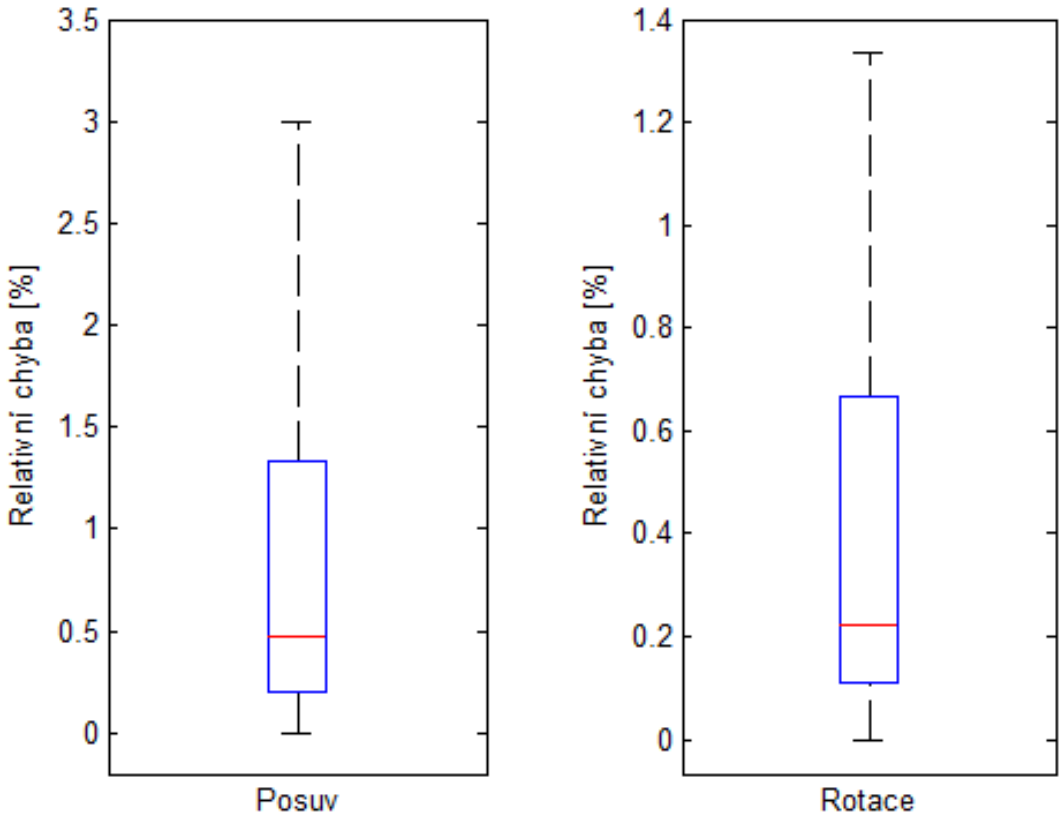
δ je relativní chyba,

A je skutečná hodnota a

γ je naměřená hodnota

Tab. 5 Stanovené chyby měření

	Δ [mm]	**δ [%]**
Posuv	$17,27 \pm 2,22$	$3,15 \pm 0,38$
Rotace	$0,32 \pm 0,06$	$0,45 \pm 0,12$

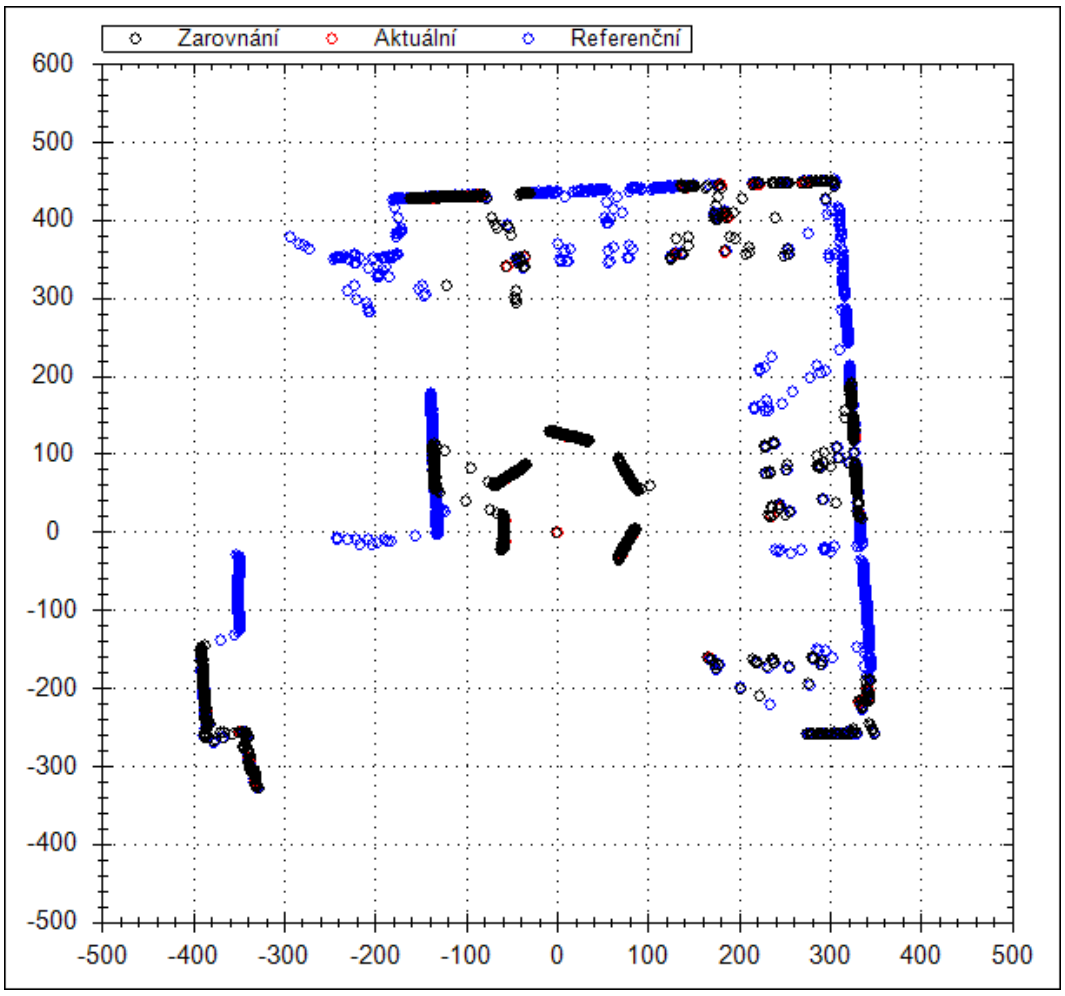


Obr. 31 Krabicové grafy relativní chyby měření posuvu a rotace

6.3 Robustnost algoritmu

Dalším důležitým parametrem algoritmu je jeho vlastní robustnost. Zmíněná robustnost je stanovena pomocí několika dynamických objektů, což jsou například osoby pohybující se po snímaném prostředí. K simulaci dynamických objektů v této diplomové práci je použito pět kartónových beden, které jsou rozmístěny okolo senzoru v určitém poloměru. Při testování robustnosti se tento poloměr rozestavění snižuje do té doby, než program vykazuje určité chyby v zarovnání aktuálního a referenčního snímku. S měřící soustavou se po nastavení referenčního snímku nehýbe a sledují se výsledné parametry zarovnání, které v ideálním případě jsou $(t_x; t_y; \phi) = (0; 0; 0)$. Určené hodnoty poloměru objektů od senzoru jsou $r = (2; 1,5; 1; 0,5)$ m. Při prvních třech hodnotách zarovnání proběhlo bez výrazných odchylek. U poloměru $r = 0,5$ m zarovnání neproběhlo úspěšně. Z tohoto důvodu je snaha najít takový poloměr

rozmístění dynamických objektů, při kterém ještě k zarovnání snímků dojde. Pomocí dalšího měření je zjištěn poloměr rozmístění objektů $r = 0,7$ m, u kterého dojde ke správnému zarovnání snímků (obr. 32). Pomocí vyhodnocovacího programu je také zjištěno, že aby došlo ke správnému zarovnání, musí mít aktuální a referenční snímek okolo 30% společných bodů.



Obr. 32 Testování robustnosti algoritmu ($r = 0,7m$)

7 Závěr

Tato diplomová práce byla zaměřena na aplikaci metody zarovnávání snímků laserového skeneru a následnou lokalizaci autonomního robotu. Nejprve byly nastudovány a teoreticky zpracovány možné metody, které jsou využitelné k lokalizaci robota v reálném prostředí. Po nastudování možných metod, kde každá z nich má své výhody i nevýhody, byl podrobněji zpracován ICP algoritmus, který je právě jednou z možností pro zarovnání snímků vytvořených laserovým senzorem a následnou lokalizací autonomního robota v reálném prostředí.

Po sepsání teoretického úvodu diplomové práce bylo jako první částí praktického zpracování vytvoření matematického modelu ICP algoritmu pomocí programu Matlab. U této realizace byl postup nejprve takový, že se k realizaci funkčního matematického modelu využívaly přídavné toolboxy softwaru. Těmito přídavnými toolboxy byly statistický a optimalizační toolbox. Po správné funkci programu byla snaha se této závislosti na přídavných funkcích zbavit. Z tohoto důvodu byly vytvořeny nové vlastní funkce nahrazující funkce z toolboxů, a nebo byla použita externí knihovna FLANN. Tato externí knihovna byla v programu použita k nalezení korespondujících bodů mezi aktuálním a referenčním snímkem.

Po zrealizování matematického modelu byl navržen program, který by byl vhodný pro online nasazení. Tato implementace algoritmu do programovacího prostředí byla provedena pomocí programovacího jazyku C#. Pomocí vývojového prostředí Microsoft Visual Studio 2010 bylo vytvořeno grafické uživatelské rozhraní. V tomto rozhraní, využívající externí knihovnu ZedGraph, bylo umožněno pracovat jak se soubory, tak pracovat s daty získanými v reálném čase pomocí laserového senzoru SICK LMS 100.

Poslední částí diplomové práce bylo otestování funkčnosti programu vytvořeného softwarem Matlab i programu vytvořeného pomocí programovacího jazyka C#. Po odzkoušení jejich funkčnosti byla snaha tyto dva programy mezi sebou porovnat. Porovnávala se hlavně jejich rychlost výpočtu a zarovnání aktuálního a referenčního snímku. Rychlost zarovnání pomocí programu vytvořeného jazykem C# byla nejméně dvakrát rychlejší, a to z důvodu paralelizace výpočtu cílové funkce. Dalším možným výsledkem diplomové práce bylo určení přesnosti lokalizace měřící soustavy při jejím pohybu po reálném prostředí. Tato přesnost byla vyjádřena pomocí určení absolutní a relativní chyby měření s 95% intervalem spolehlivosti jak v translačním pohybu, tak i v pohybu rotačním. U translačního pohybu však vznikla integrační chyba, a to z důvodu znovunastavení referenčního snímku při posuvu měřící soustavy

o vzdálenost $1,5$ m. Posledním výsledkem bylo odzkoušení robustnosti algoritmu. Tato robustnost byla ověřena použitím dynamických objektů, které byly simulovány pomocí několika kartonových krabic. U této robustnosti se zjistilo, že algoritmus pracuje správně do té doby, dokud objekty rozložené okolo senzoru jsou vzdáleny minimálně v poloměru 70 cm a snímky mají alespoň 30% bodů shodných.

Další možnou budoucí prací vycházející z této diplomové práce by mohlo být nahrazení FLANN knihovny pro nalezení nejbližších sousedů mezi snímky vlastní knihovnou. Následujícím krokem by mohlo být urychlení programu jeho optimalizací, hlavně v počtu účelových funkcí s různými počátečními podmínkami a hodnotami těchto podmínek. Zmíněna optimalizace programu by také mohla minimalizovat integrační chybu, která vznikla při posuvu měřící soustavy.

8 Použitá literatura

[1] BIBER, P., STRASSER, W. The Normal Distributions Transform: A new approach to laser scan matching. *In Proc. of the 2003 IEEE International Conference on Intelligent robots and Systems*, pp. 2746-2748, 2003

[2] BESL, P., MCKAY, N. A Method for Registration of 3-D Shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (Los Alamitos, CA, USA: IEEE Computer Society)*,Vol.17, No.8, pp. 239-256, 1992

[3] RUSINKIEWICZ, S., LEVOY, M. Efficient variants of the ICP algorithm. *In Proceedings of Third International Conference on 3-D Digital Imaging and Modeling*, Quebec City, Que, 28 May 2001-01 Jun 2001. pp.145–152. ISBN 0-7695-0984-3.

[4] TAKUBO, T., KAMINADE, T. NDT scan matching method for high resolution grid map. *In Proc. of the 2009 IEEE International Conference on Intelligent robots and Systems*, St. Louis, USA, October 11-15 2009

[5] WEBER, J., JÖRG, K., PUTTKAMER, E., APR - Global scan matching using anchor point relationships. *In Proc. of the 6th international conference on intelligent autonomous systems*, IOS press, pp. 471-478, 2000

[6] ZEZHONG, X., JILIN, L., ZHIYU, X., Scan matching based on CLS relationship. *In Proc. of International Conference on Robotics, Inteligent Systems and Signal Processing*, Gangsha, China, October 2003

[7] CENSI, A., GRISETTI, G., Scan Matching in the Hough Domain. *in Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005

[8] LEE, H., LEE, S.,LEE, S., Comparation and Analysis of Scan Matching Techniques of Cooperative - SLAM. *in Proceedings of the 8th International International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Incheon, Korea, November 23-26. 2011

[9] WEISS, G., WETZLER, C., von PUTTKAMER, E. Keeping Track of Position and Orientation of Moving Indoor Systems by Correlation of Range-Finder Scans. *In: Proc.Int. Conf. on Intelligent Robots and Systems (IROS-94)*. Munich, Germany, 595-601, 1994 .

[10] QUI, Q., HAN, J., Laser Scan Matching Ssing Multiplex Histograms with Feature Components. *In: Proc.Int. Conf. on Robotics and Biometics*. Guilin, China, December 19-23. 2009.

[11] RÖFER, T., Using Histogram Correlation to Create Consistent Laser Scan Maps. *In: Proc.Int. Conf. on Robotics Systems (IROS-2002)*. Lausanne, Switzerland, 625-630, 2002.

[12] DIOSI, A., KLEEMAN, L., Fast Laser Scan Matching using Polar Coordinates. *The International Journal of Robotics Research*. Vol. 26, No. 10, pp. 1125-1153, October 2007.

[13] HANDSCHIN, J. Monte Carlo techniques for prediction and filtering. Auto - matica, sv. 6, p. 555–563, 1970.

[14] FOX, D., WOLFRAM, B., DELLAERT, F., THURN, S. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *v Proceedings of the Sixteenth National Conference on Artificial Intelligence*. Orlando, USA, 1999.

[15] FOX, D., WOLFRAM, B., DELLAERT, F., THRUN, S. Robust Monte Carlo Localization for Mobile Robots. *Computer Science Department*. University of Freiburg, Germany, 2001.

[16] SKANSKÝ, M. Lokalizace robota v podmínkách soutěže Eurobot. *Bakalářská práce*. Masarykova Univerzita, Fakulta informatiky, Brno, 2012.

[17] KJER, H., WILM, J., Evaluation of surface registration algorithms for PET motion correction. *Bakalářská práce*. Technical University of Denmark, Informatics and Mathematical Modeling, Denmark, 2010.

[18] NOCEDAL, J., WRIGHT, S., Numerical Optimalization. 3.vyd. New York: Springer, 1999. ISBN: 0-387-98793-2(HC)

[19] MACHALOVA, J., NEKUTA, H. Numerické metody nepodmíněné optimalizace. Leden 2013, Olomouc.

[20] MUJA, M., LOWE, D., FLANN - Fast Library for Approximate Nearest Neighbors. *User Manual* January 24,2013.

[21] DOSTÁL, Z. Lineární Algebra. 2000, Ostrava.

[22] LITCHMANNOVÁ, M. Úvod do statistiky. 2011, Ostrava.

[23] KABELOVÁ, A., DOSTÁLEK, L. Velký průvodce protokoly TCP/IP a systémem DNS. 2008, Praha.

[24] Uživatelská příručka SICK. SICK LMS100/111/120. [online] 2014. [cit. 2014-3-15]. Dostupné z: http://www.sick-automation.ru/images/File/pdf/DIV05/LMS100_manual.pdf.

[25] SOURCEFORGE.NET - ZedGraph library. [online] 2014. [cit. 2014-2-20]. Dostupné z: http://sourceforge.net/projects/zedgraph/.